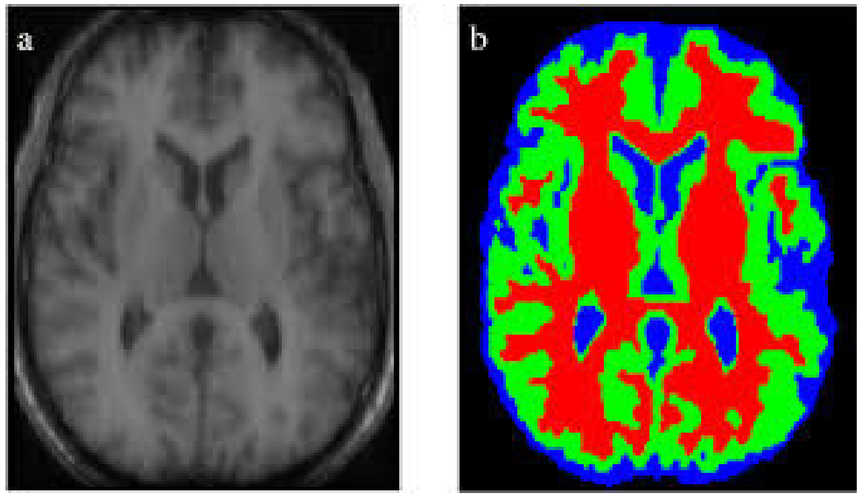
Semantic Segmentation Networks

What is semantic segmentation?

Classifying each pixel of an image into a class (a pre-defined category)

Semantic segmentation in
autonomous driving



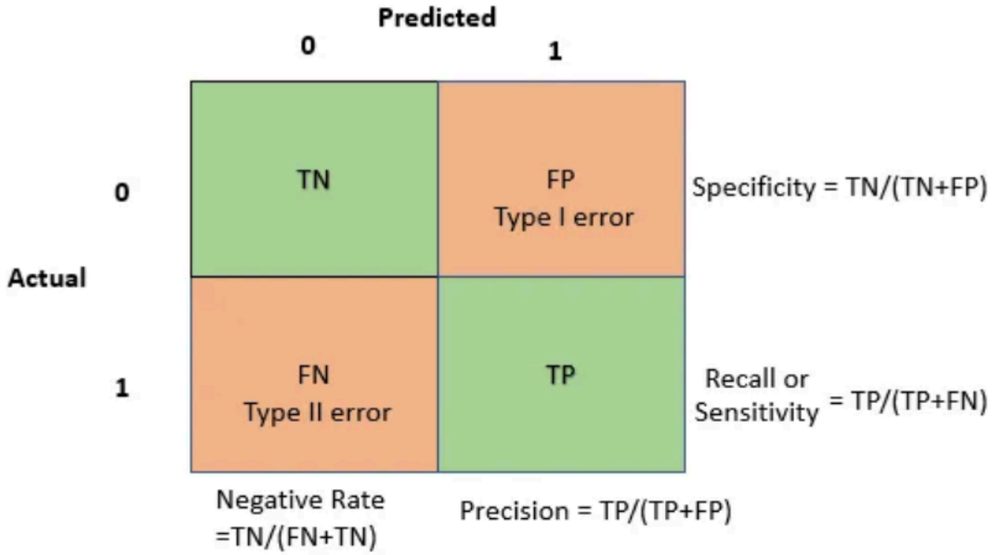
Semantic segmentation in
medical imaging

Evaluation Metrics

Methods to quantify the results for easy evaluation and comparison

- Confusion Matrix -
  - A confusion matrix is an N X N matrix, where N is the number of predicted classes.
  - It is a performance measurement for classification problems where the output can be two or more classes.
  - Specifically, it’s a table with different combinations of predicted (by our model) and actual values (ground truth).
    - True Positive: You predicted positive, and it’s true.
    - True Negative: You predicted negative, and it’s true.
    - False Positive: (Type 1 Error): You predicted positive, and it’s false.
    - False Negative: (Type 2 Error): You predicted negative, and it’s false.

- Accuracy
  - = (TP + TN)/(TP + TN + FP + FN)
  - Correct Predictions / All predictions
  - Measures how often a machine learning model correctly predicts the outcome.
- Precision
  - = (TP)/(TP + FP)
  - Measures how often a machine learning model correctly predicts the positive class.
- Recall
  - = (TP)/(TP + FN)
  - = Sensitivity
  - Measures how often a machine learning model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset.
- F1 Score OR Dice Score
  - The harmonic mean of precision and recall, and is used to balance the trade-off between precision and recall.
  - $= \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} * 100\%$



$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

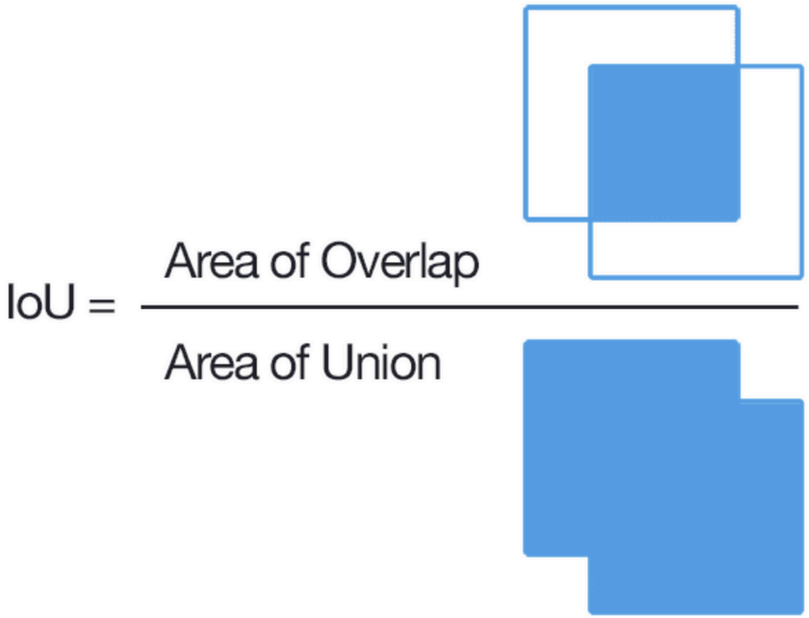
$$\text{F1 - Score} = \frac{2*\text{Recall}*\text{Precision}}{\text{Recall} + \text{Precision}}$$

Source

Intersection over Union (IoU)

- Intersection over Union (IoU) is a metric used to evaluate the accuracy of object detection models, such as object localization and image segmentation.
- It measures the overlap between two bounding boxes: the predicted bounding box and the ground truth bounding box.
- An IoU threshold (e.g., 0.5) is often set to determine if a predicted bounding box is a true positive or a false positive. If IoU > threshold, the prediction is considered accurate.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Source

Fully Convolutional Networks

- Authors of “FCN for Semantic Segmentation” show that a fully convolutional network (FCN), trained end-to-end, pixels-to-pixels on semantic segmentation exceeds the state-of-the-art without further machinery.
- This is the first work to train FCNs end-to-end
  - (1) for pixel-wise prediction and
  - (2) from supervised pre-training.
- Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs.
- Both learning and inference are performed whole-image at a time by dense feedforward computation and backpropagation.
- In-network upsampling layers enable pixel-wise prediction and learning in nets with subsampled pooling.
- This model transferred recent success in classification to a dense prediction by reinterpreting classification nets as fully convolutional and fine-tuning from their learned representations.

Fully Convolutional Networks for Semantic Segmentation
Jonathan Long, Evan Shelhamer, Trevor Darrell

- Semantic segmentation faces an inherent tension between semantics and location: global information resolves “what” while local information resolves “where”.
- Deep feature hierarchies jointly encode location and semantics in a local-to global-pyramid.
- The authors define a novel “skip” architecture to combine deep, coarse, semantic information and shallow, fine, appearance information.

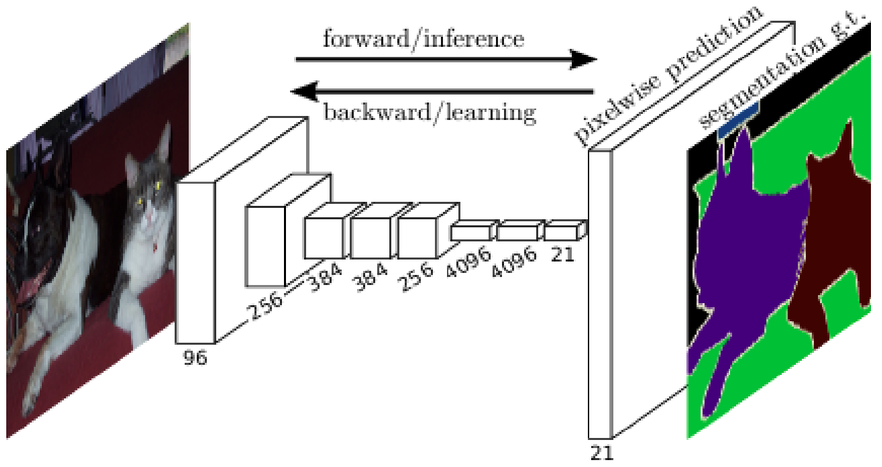


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Segmentation Architecture

- We cast ILSVRC (ImageNet Challenge) classifiers into FCNs and augment them for dense prediction with in-network upsampling and a pixel-wise loss.
- Then, train for segmentation by fine-tuning.
- Next, build a novel skip architecture that combines coarse, semantic and local, appearance information to refine prediction.
- We train with a per-pixel multinomial logistic loss and validate with the standard metric of mean pixel intersection over union (IoU), with the mean taken over all classes, including background.
- The training ignores pixels that are masked out (as ambiguous or difficult) in the ground truth.

From Classifiers to Dense FCNs

- The authors convolutionalize proven classification architectures like:
  - AlexNet (ILSVRC12 winner)
  - VGG-16
  - GoogLeNet (ILSVRC14 winner)
- They decapitate each net by discarding the final classifier layer, and convert all fully connected layers to convolutions.
- A 1 x 1 convolution is appended with channel dimension 21 to predict scores for each of the PASCAL classes (including background) at each of the coarse output locations, followed by a deconvolution layer to bi-linearly upsample the coarse outputs to pixel-dense outputs.

- Fine-tuning from classification to segmentation gives a further boost in performance.
- Even the worst model achieved 75% of state-of-the-art performance.
- The segmentation-equipped VGG net (FCN-VGG16) already appears to be state-of-the-art at 56.0 mean IoU on the validation set, compared to 52.6 on the test set.
- Training on extra data raises performance to 59.4 mean IoU on a subset of validation set.

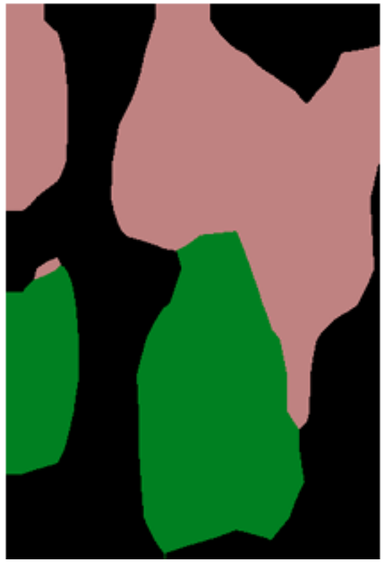
	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

Table 1. We adapt and extend three classification convnets to segmentation. We compare performance by mean intersection over union on the validation set of PASCAL VOC 2011 and by inference time (averaged over 20 trials for a 500×500 input on an NVIDIA Tesla K40c). We detail the architecture of the adapted nets as regards dense prediction: number of parameter layers, receptive field size of output units, and the coarsest stride within the net. (These numbers give the best performance obtained at a fixed learning rate, not best performance possible.)

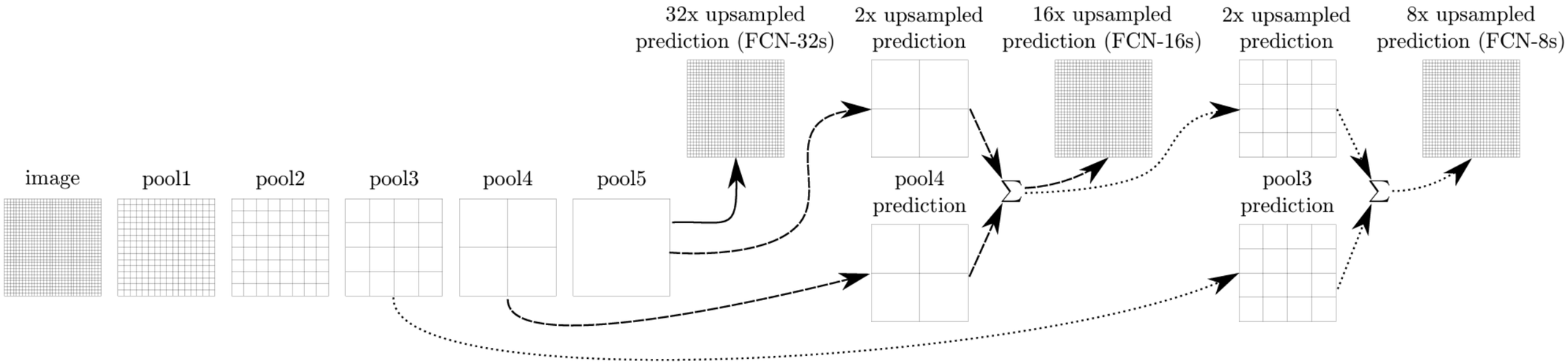
Skip Connections

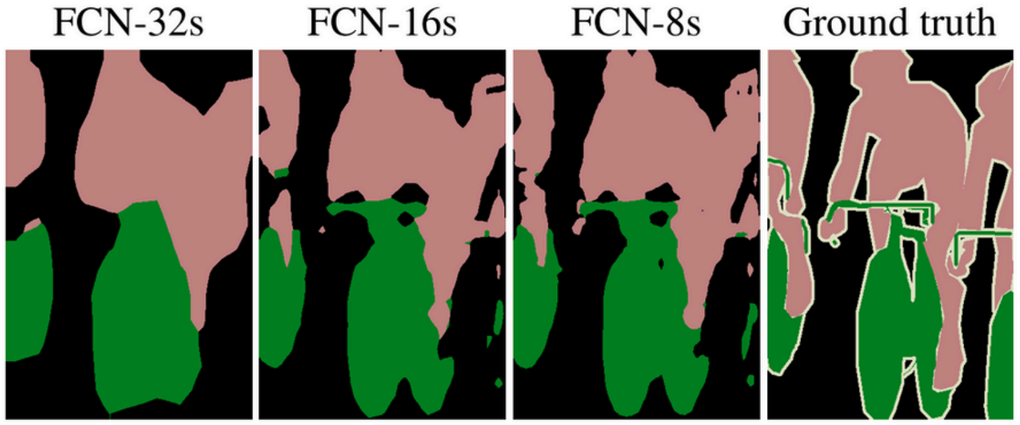
- The authors define a new fully convolutional net (FCN) for seg mentation that combines layers of the feature hierarchy and refines the spatial precision of the output.
- While fully convolutionalized classifiers can be fine-tuned to segmentation as shown in the previous slides, and even score highly on the standard metric, their output is dissatisfyingly coarse.

- This issue is addressed by adding links that combine the final prediction layer with lower layers with finer strides.
- This turns a line topology into a DAG, with edges that skip ahead from lower layers to higher ones.
- As they see fewer pixels, the finer scale predictions should need fewer layers, so it makes sense to make them from shallower net outputs.
- Combining fine layers and coarse layers lets the model make local predictions that respect global structure.



Without using skip connections





Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail.

- We first divide the output stride in half by predicting from a 16 pixel stride layer.
- We add a 1 x 1 convolution layer on top of pool4 to produce additional class predictions. We fuse this output with the predictions computed on top of conv7 (convolutionalized fc7) at stride 32 by adding a 2x upsampling layer and summing both predictions.
- We initialize the 2x upsampling to bilinear interpolation but allow the parameters to be learned.
- Finally, the stride 16 predictions are upsampled back to the image.
- We call this net FCN-16s. FCN-16s is learned end-to-end, initialized with the parameters of the last, coarser net, which we now call FCN-32s.
- The new parameters acting on pool4 are zero-initialized, so the net starts with unmodified predictions.
- The learning rate is decreased by a factor of 100.
- **Learning this skip net improves performance on the validation set by 3.0 mean IoU to 62.4.**
- We continue in this fashion by fusing predictions from pool3 with a 2x upsampling of predictions fused from pool4 and conv7, building the net FCN-8s.

Table 2. Comparison of skip FCNs on a subset of PASCAL VOC2011 validation[7]. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

U-Net

- The authors of U-Net build upon the Fully Convolutional Network (FCN) and modify and extend this architecture to work with very few training images and yield more precise segmentations.
- The main idea in FCN is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators.
- Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled output.
- A successive convolution layer can then learn to assemble a more precise output based on this information.
- One important modification in U-Net is that in the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. Therefore, the expansive path is more or less symmetric to the contracting path, and yields a U-shaped architecture.
- The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image.

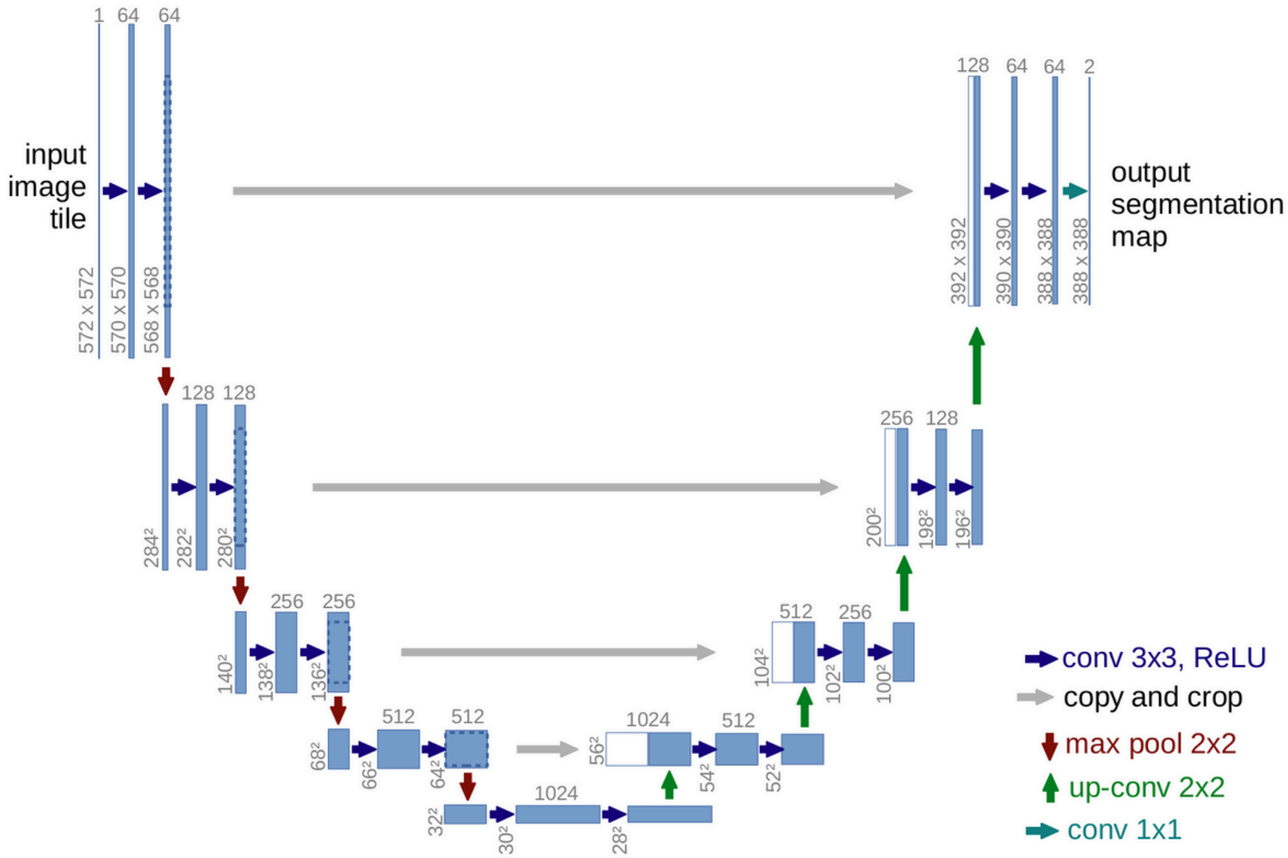


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

U-Net: Convolutional Networks for Biomedical Image Segmentation
Olaf Ronneberger, Philipp Fischer, Thomas Brox

- To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory.

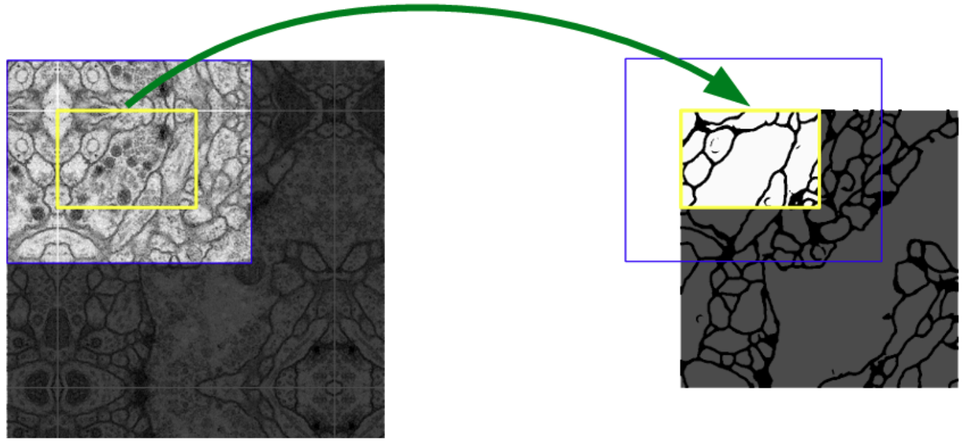


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

- U-Net is specifically aimed at solving tasks concerned with biomedical image processing and for such tasks there is often very little training data available as creating labeled data is expensive and time consuming.
- Therefore, we use excessive data augmentation by applying elastic deformations to the available training images. This allows the network to learn invariance to such deformations without the need to see these transformations in the annotated image corpus.
- This is particularly important in biomedical segmentation, since deformation is the most common variation in tissue and realistic deformations can be simulated efficiently.
- Another challenge in many cell segmentation tasks is the separation of touching objects of the same class. For this, the authors propose the use of a weighted loss, where the separating background labels between touching cells obtain a large weight in the loss function.
- The resulting network is applicable to various biomedical segmentation problems.

Network Architecture

- The architecture consists of a contracting path (left side) and an expansive path (right side).
- The contracting path follows the typical architecture of a convolutional network.
- It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling.
- At each downsampling step we double the number of feature channels.

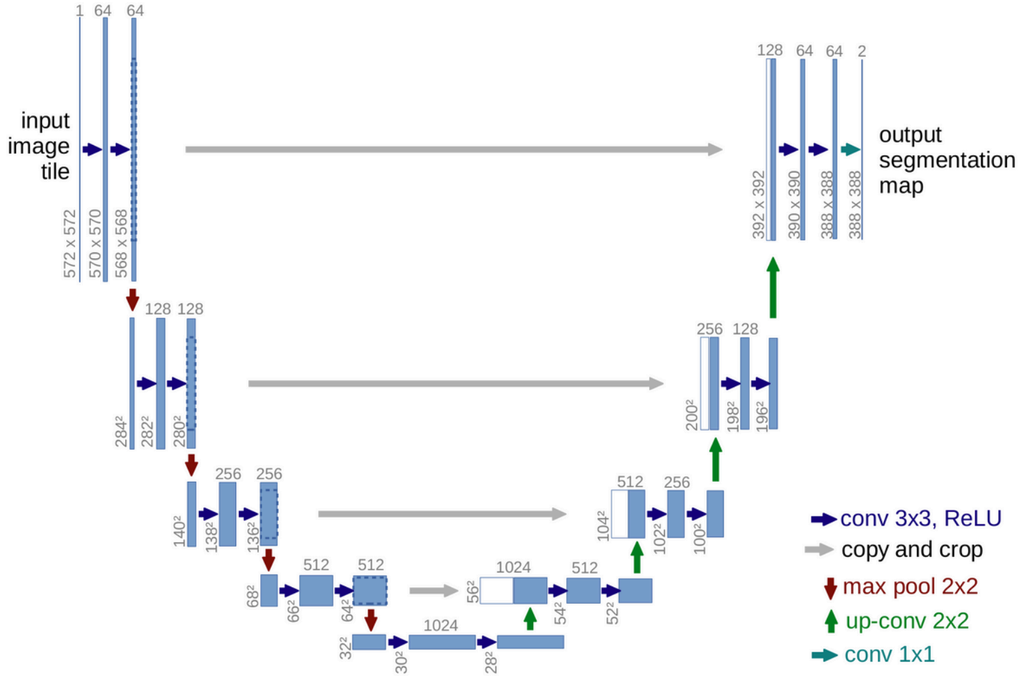


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

- Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution (up-convolution) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.
- The cropping is necessary due to the loss of border pixels in every convolution.
- At the final layer a 1x1 convolution is used to map each 64 component feature vector to the desired number of classes.
- In total the network has 23 convolutional layers.

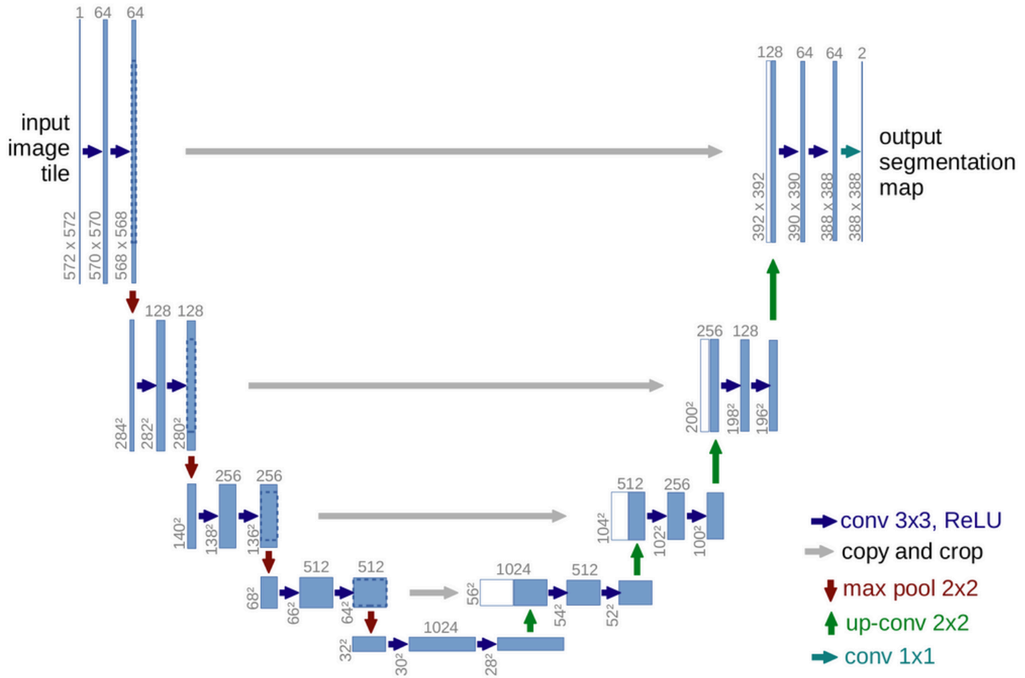


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Training

- The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent implementation of Caffe.
- Due to the unpadded convolutions, the output image is smaller than the input by a constant border width.
- The energy function is computed by a pixel-wise soft-max over the nal feature map combined with the cross entropy loss function.
- The soft-max is defined as - $\exp(a_k(\mathbf{x}))/\left(\sum_{k'=1}^{K}\exp(a_{k'}(\mathbf{x}))\right)$
- Here ak(x) denotes the activation in feature channel k at the pixel position x.
- K is the number of classes and pk(x) is the approximated maximum-function. I.e. pk(x) is close to 1 for the k that has the maximum activation ak(x) and pk(x) is close to 0 for all other k.
- The cross entropy then penalizes at each position the deviation of $p_{\ell(\mathbf{x})}(\mathbf{x})$ from 1 using $E=\sum_{\mathbf{x}\in\Omega} w(\mathbf{x})\log(p_{\ell(\mathbf{x})}(\mathbf{x}))$

where $\ell : \Omega \to \{1,\dots,K\}$ is the true label of each pixel and $w : \Omega \to \mathbb{R}$ is a weight map that we introduced to give some pixels more importance in the training.

Data Augmentation

- Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available.
- In case of microscopical images we primarily need
  - shift and rotation invariance, as well as
  - robustness to deformations and gray value variations.
- Especially random elastic deformations of the training samples seem to be the key concept to train a segmentation network with very few annotated images.
- We generate smooth deformations using random displacement vectors on a coarse 3 by 3 grid.
- The displacements are sampled from a Gaussian distribution with 10 pixels standard deviation.
- Per-pixel displacements are then computed using bicubic interpolation.
- Drop-out layers at the end of the contracting path perform further implicit data augmentation.

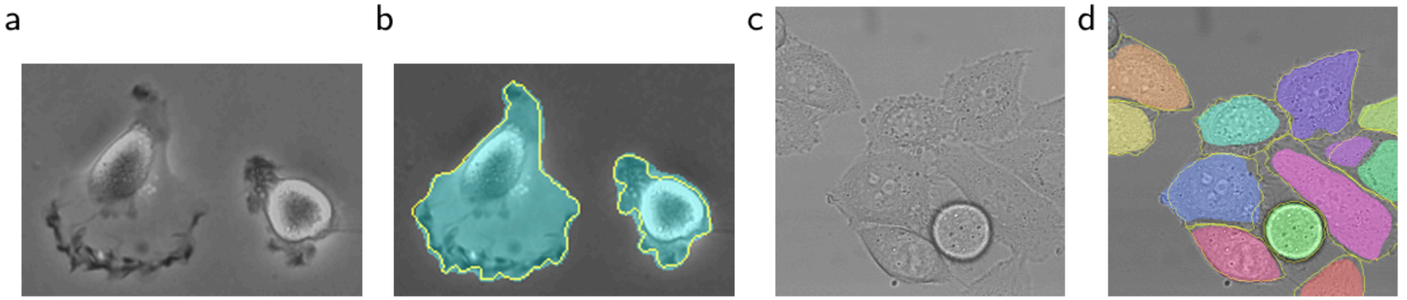


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

Table 2. Segmentation results (IOU) on the ISBI cell tracking challenge 2015.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

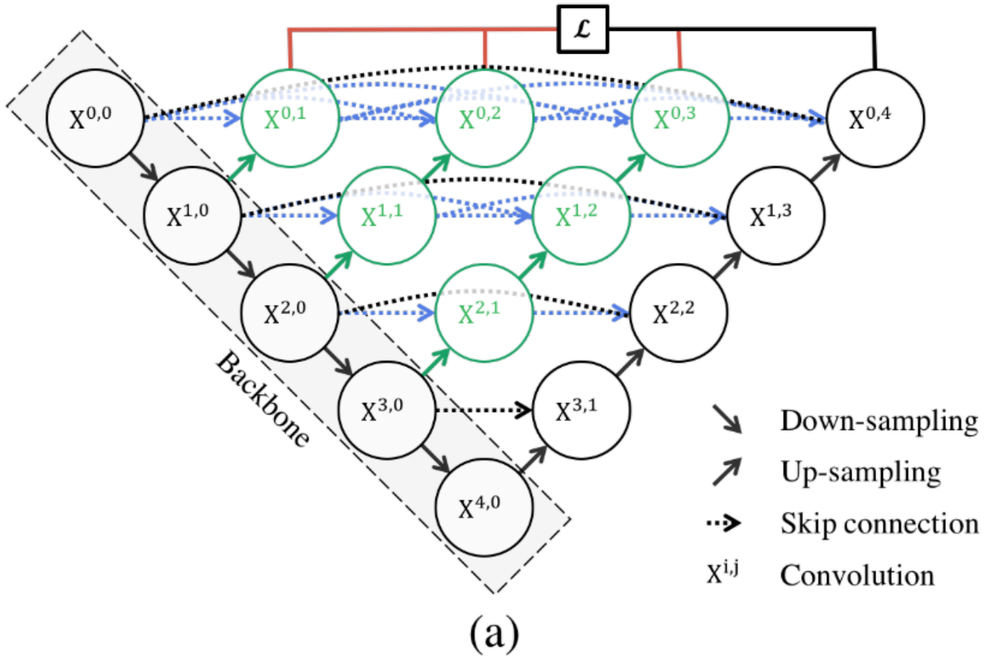
U-Net ++

- UNet++ is a new segmentation architecture based on nested and dense skip connections.
- The underlying hypothesis behind this architecture is that the model can more effectively capture fine-grained details of the foreground objects when high-resolution feature maps from the encoder network are gradually enriched before fusion with the corresponding semantically rich feature maps from the decoder network.
- The authors argue that the network would deal with an easier learning task when the feature maps from the decoder and encoder networks are semantically similar.
- This is in contrast to the plain skip connections commonly used in U-Net, which directly fast-forward high-resolution feature maps from the encoder to the decoder network, resulting in the fusion of semantically dissimilar feature maps.

UNet++: A Nested U-Net Architecture for Medical Image Segmentation
Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang

Proposed U-Net++ Architecture

- U-Net++ starts with an encoder sub-network or backbone, followed by a decoder sub-network.
- What distinguishes UNet++ from U-Net (the black components) is the re-designed skip pathways (in green and blue) that connect the two sub-networks and the use of deep supervision (shown red).



(a)

Re-designed skip pathways

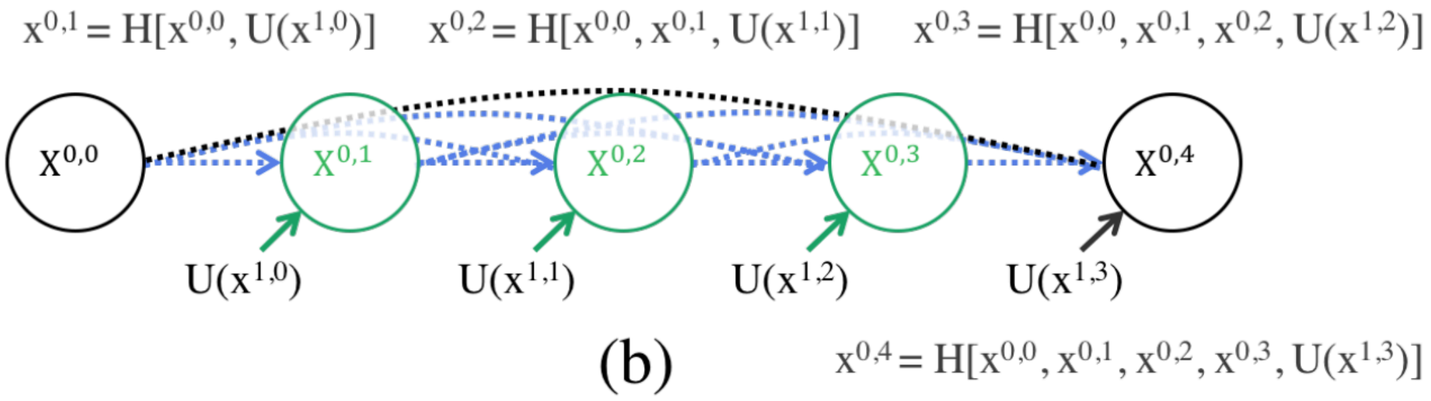
- Re-designed skip pathways transform the connectivity of the encoder and decoder sub-networks.
- In U-Net, the feature maps of the encoder are directly received in the decoder; however, in UNet++, they undergo a dense convolution block whose number of convolution layers depends on the pyramid level.
- For example, the skip pathway between nodes X(0,0) and X(1,3) consists of a dense convolution block with three convolution layers where each convolution layer is preceded by a concatenation layer that fuses the output from the previous convolution layer of the same dense block with the corresponding up-sampled output of the lower dense block.
- Essentially, the dense convolution block brings the semantic level of the encoder feature maps closer to that of the feature maps awaiting in the decoder.
- The hypothesis is that the optimizer would face an easier optimization problem when the received encoder feature maps and the corresponding decoder feature maps are semantically similar.

Formally, we formulate the skip pathway as follows: let $x^{i,j}$ denote the output of node $\mathrm{X}^{i,j}$ where i indexes the down-sampling layer along the encoder and j indexes the convolution layer of the dense block along the skip pathway. The stack of feature maps represented by $x^{i,j}$ is computed as

$$x^{i,j} = \begin{cases} \mathcal{H}\left(x^{i-1,j}\right), & j = 0 \\ \mathcal{H}\left(\left[\left[x^{i,k}\right]_{k=0}^{j-1}, \mathcal{U}(x^{i+1,j-1})\right]\right), & j > 0 \end{cases} \tag{1}$$

where function $\mathcal{H}(\cdot)$ is a convolution operation followed by an activation function, $\mathcal{U}(\cdot)$ denotes an up-sampling layer, and [ ] denotes the concatenation layer.

- Basically, nodes at level j = 0 receive only one input from the previous layer of the encoder; nodes at level j = 1 receive two inputs, both from the encoder sub-network but at two consecutive levels; and nodes at level j > 1 receive j + 1 inputs, of which j inputs are the outputs of the previous j nodes in the same skip pathway and the last input is the up-sampled output from the lower skip pathway.



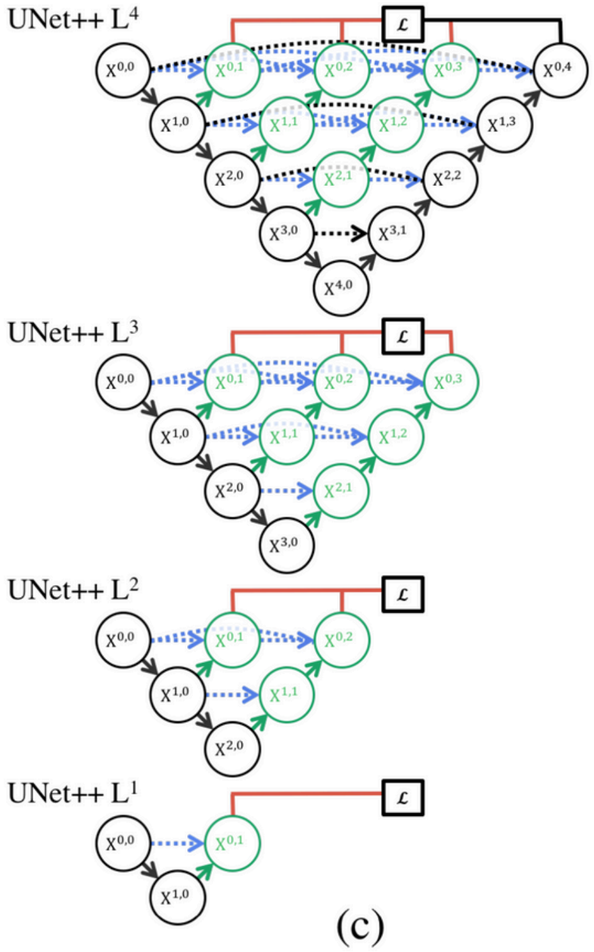
Detailed analysis of the first skip pathway of UNet++

Deep Supervision

- The authors propose to use deep supervision in UNet++, enabling the model to operate in two modes:-
  a. Accurate mode wherein the outputs from all segmentation branches are averaged.
  b. Fast mode wherein the final segmentation map is selected from only one of the segmentation branches, the choice of which determines the extent of model pruning and speed gain.
- Owing to the nested skip pathways, UNet++ generates full resolution feature maps at multiple semantic levels, {x(0,j) , j ∈ {1, 2, 3, 4}}, which are amenable to deep supervision.
- The authors have added a combination of binary cross-entropy and dice coefficient as the loss function to each of the above four semantic levels, which is described as:-

$$\mathcal{L}(Y, \hat{Y}) = -\frac{1}{N}\sum_{b=1}^{N}\left(\frac{1}{2}\cdot Y_b \cdot \log \hat{Y}_b + \frac{2\cdot Y_b \cdot \hat{Y}_b}{Y_b + \hat{Y}_b}\right) \tag{2}$$

where $\hat{Y}_b$ and Y_b denote the flatten predicted probabilities and the flatten ground truths of b^{th} image respectively, and N indicates the batch size.



UNet++ can be pruned at inference time, if trained with deep supervision.

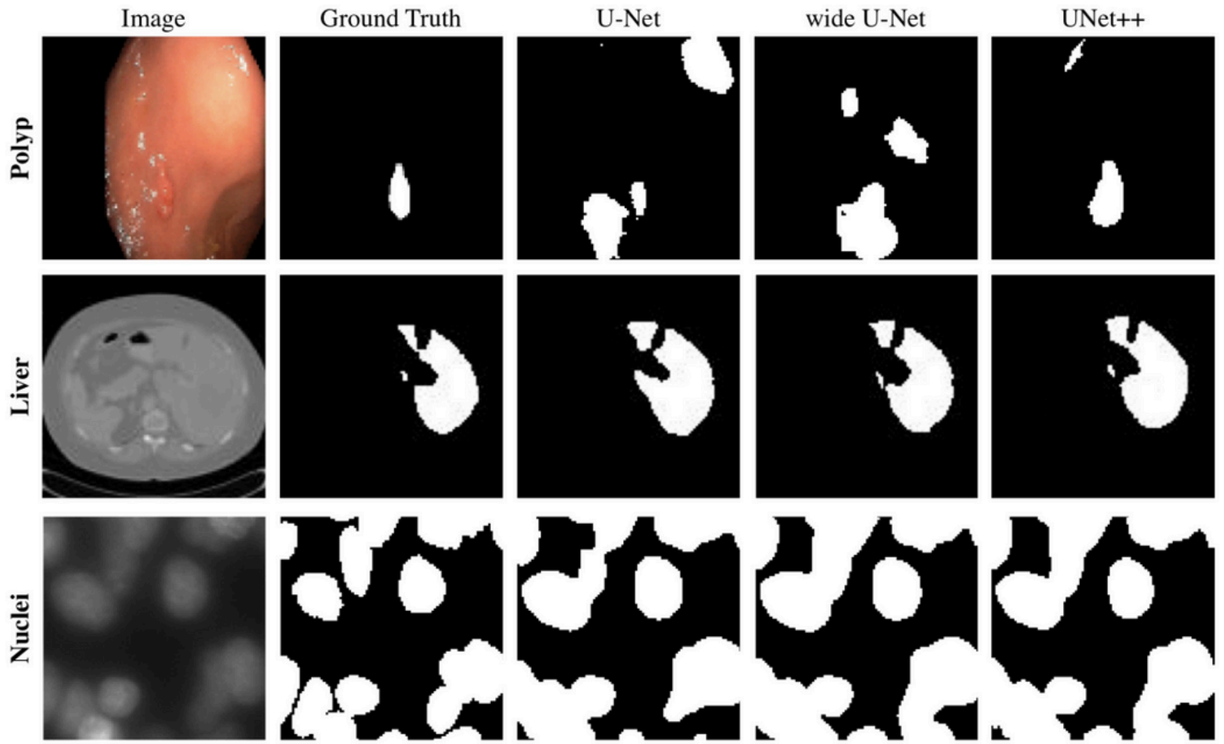
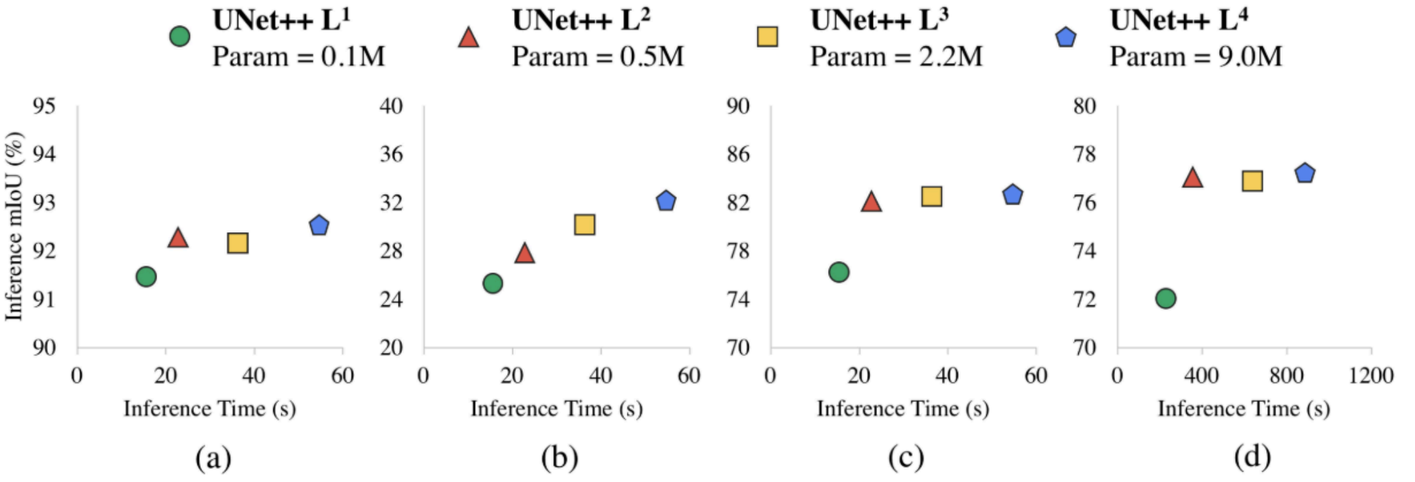


Fig. 2: Qualitative comparison between U-Net, wide U-Net, and UNet++, showing segmentation results for polyp, liver, and cell nuclei datasets (2D-only for a distinct visualization).

Table 3: Segmentation results (IoU: %) for U-Net, wide U-Net and our suggested architecture UNet++ with and without deep supervision (DS).

Architecture	Params	Dataset			
		cell nuclei	colon polyp	liver	lung nodule
U-Net [9]	7.76M	90.77	30.08	76.62	71.47
Wide U-Net	9.13M	90.92	30.14	76.58	73.38
UNet++ w/o DS	9.04M	**92.63**	**33.45**	79.70	76.44
UNet++ w/ DS	9.04M	92.52	32.12	**82.90**	**77.21**



Applying different levels of pruning

Fig. 3: Complexity, speed, and accuracy of UNet++ after pruning on (a) cell nuclei, (b) colon polyp, (c) liver, and (d) lung nodule segmentation tasks respectively. The inference time is the time taken to process **10k** test images using one NVIDIA TITAN X (Pascal) with 12 GB memory.

Motivation for SegNet

- It is primarily motivated by road scene understanding applications that require the ability to model appearance (road, building), shape (cars, pedestrians) and understand the spatial-relationship (context) between different classes such as road and side-walk.
- In typical road scenes, the majority of the pixels belong to large classes such as road, building and hence the network must produce smooth segmentations.
- The engine must also have the ability to delineate objects based on their shape despite their small size.
- Hence it is important to retain boundary information in the extracted image representation.
- From a computational perspective, it is necessary for the network to be efficient in terms of both memory and computation time during inference.
- The ability to train end-to-end in order to jointly optimise all the weights in the network using an efficient weight update technique such as stochastic gradient descent (SGD)

SegNet

- The encoder network in SegNet is topologically identical to the convolutional layers in VGG16.
- We remove the fully connected layers of VGG16 which makes the SegNet encoder network significantly smaller and easier to train than many other recent architectures (like FCN).
- The key component of SegNet is the decoder network which consists of a hierarchy of decoders one corresponding to each encoder.
- Of these, the appropriate decoders use the max-pooling indices received from the corresponding encoder to perform non-linear upsampling of their input feature maps.
- Reusing max-pooling indices in the decoding process has several practical advantages :-
    - It improves boundary delineation
    - It reduces the number of parameters enabling end-to-end training.
    - This form of upsampling can be incorporated into any encoder-decoder architecture (such as FCN) with slight modification.

Architecture

- SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer.
- The encoder network consists of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG16 network designed for object classification.
- We can therefore initialize the training process from weights trained for classification on large datasets.
- We can also discard the fully connected layers in favour of retaining higher resolution feature maps at the deepest encoder output. This also reduces the number of parameters in the SegNet encoder network significantly (from 134M to 14.7M) as compared to other recent architectures.
- Each encoder layer has a corresponding decoder layer and hence the decoder network has 13 layers.
- The final decoder output is fed to a multi-class soft-max classifier to produce class probabilities for each pixel independently.

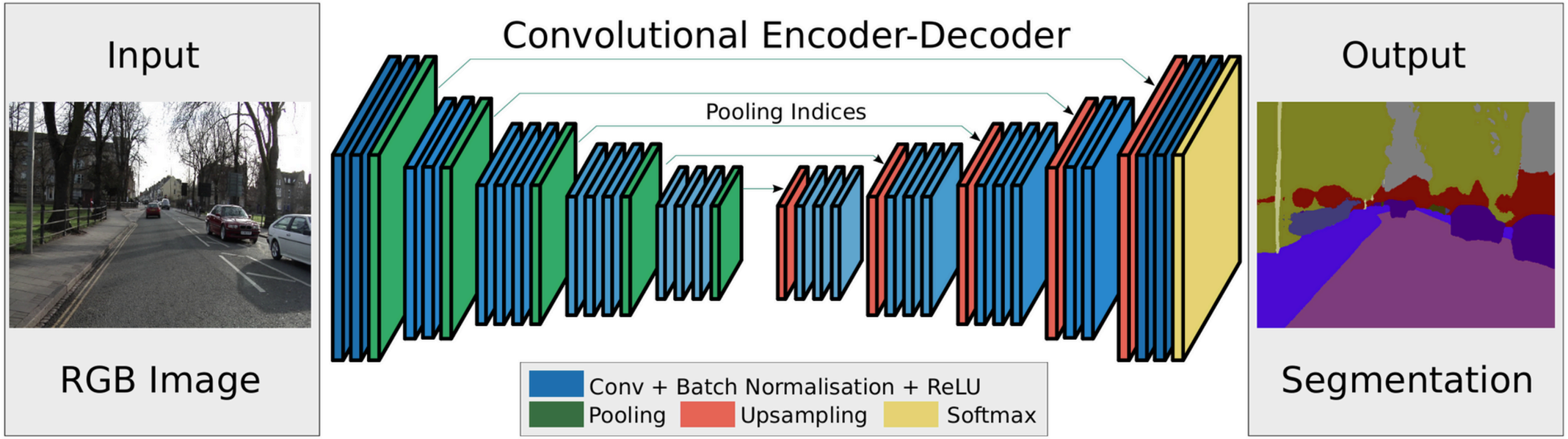


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation
Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla

Encoders

- Each encoder in the encoder network performs convolution with a filter bank to produce a set of feature maps.
- These are then batch normalized.
- Then an element-wise rectified linear non-linearity (ReLU) max(0,x) is applied.
- Following that, max-pooling with a 2 x 2 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2.
- Max-pooling is used to achieve translation invariance over small spatial shifts in the input image.
- Sub-sampling results in a large input image context (spatial window) for each pixel in the feature map.
- While several layers of max-pooling and sub-sampling can achieve more translation invariance for robust classification correspondingly there is a loss of spatial resolution of the feature maps.
- The increasingly lossy (boundary detail) image representation is not beneficial for segmentation where boundary delineation is vital.
- Therefore, it is necessary to capture and store boundary information in the encoder feature maps before sub-sampling is performed.

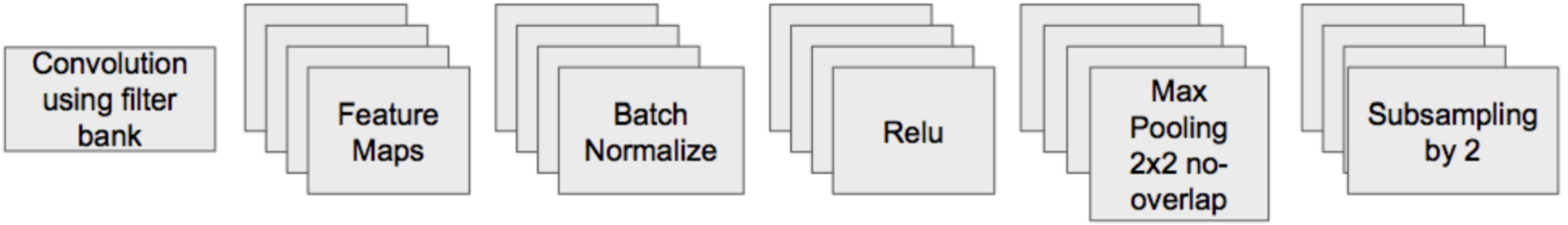
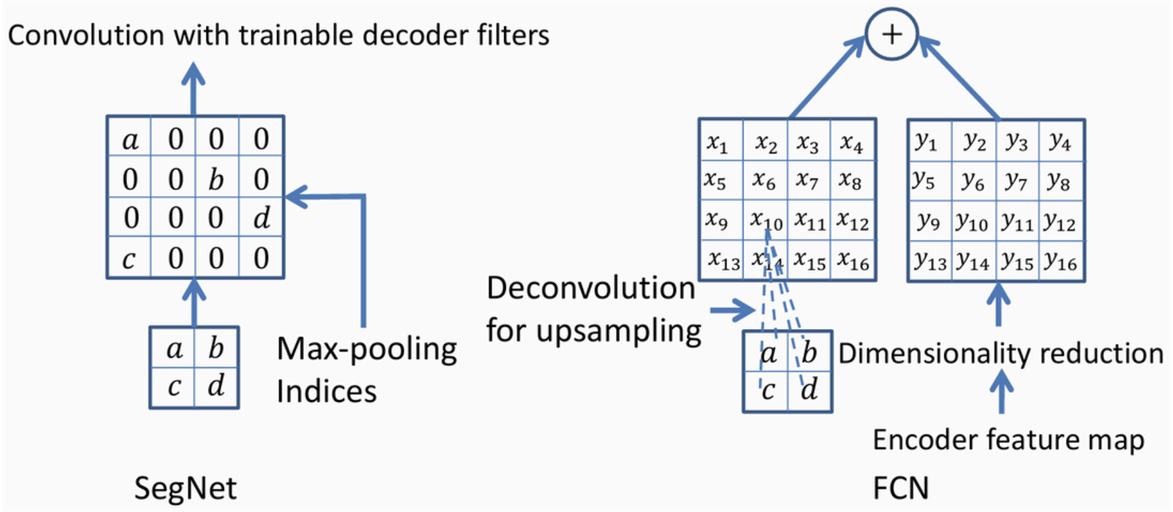


**Image from lecture slides of Jianping Fan, UNC-Charlotte*

- If memory during inference is not constrained, then all the encoder feature maps (after sub sampling) can be stored but this is usually not the case in practical applications and hence there needs to be a more efficient way to store this information.
- There is such a way and it involves storing only the max-pooling indices, i.e, the locations of the maximum feature value in each pooling window is memorized for each encoder feature map.
- In principle, this can be done using 2 bits for each 2 x 2 pooling window and is thus much more efficient to store as compared to memorizing feature map(s) in float precision.
- This lower memory storage results in a slight loss of accuracy but is still suitable for practical applications.



- The appropriate decoder in the decoder network upsamples its input feature map(s) using the memorized max-pooling indices from the corresponding encoder feature map(s).
- This step produces sparse feature map(s). This SegNet decoding technique is illustrated in the previous slide.
- These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps.
- A batch normalization step is then applied to each of these maps.
- Note that the decoder corresponding to the first encoder (closest to the input image) produces a multi-channel feature map, although its encoder input has 3 channels (RGB). This is unlike the other decoders in the network which produce feature maps with the same number of size and channels as their encoder inputs.
- The high dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier. This soft-max classifies each pixel independently.
- The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel.

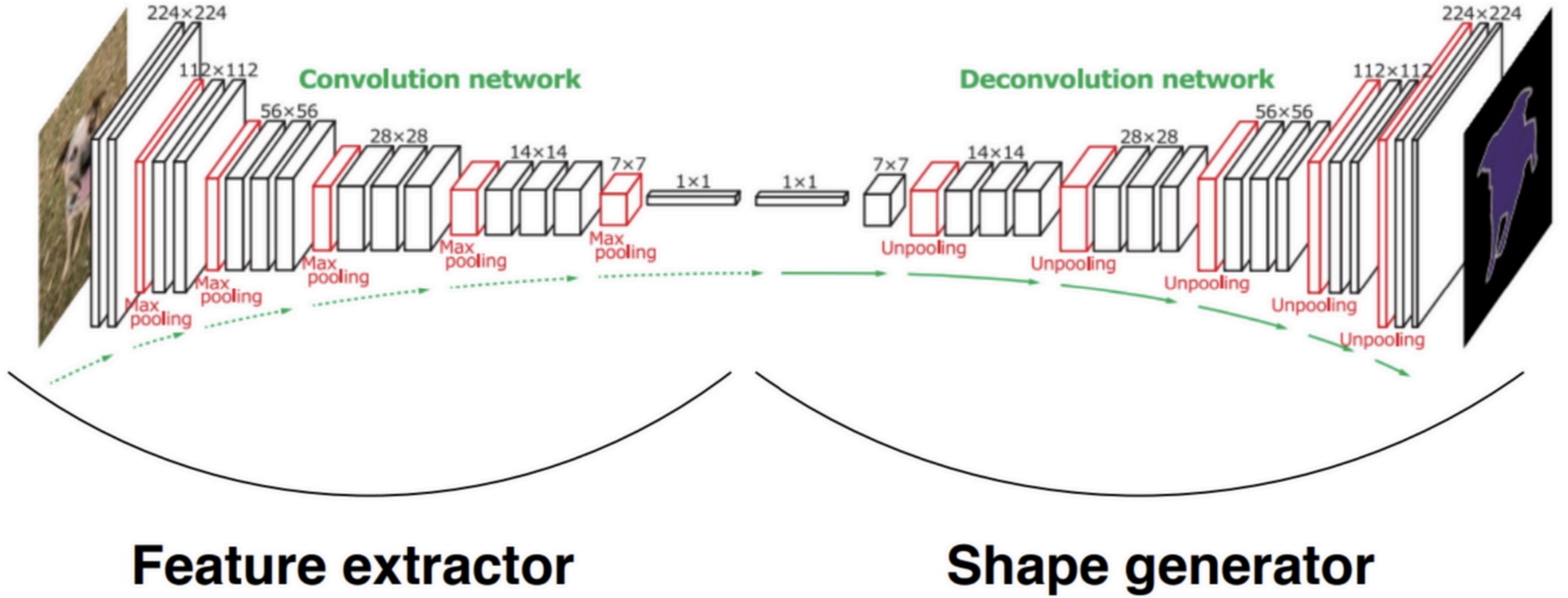


**Image credit to Gregory P. Spell*

U-Net and SegNet

- U-Net shares a similar architecture to SegNet but with some differences.
- As compared to SegNet, U-Net (proposed for the medical imaging community) does not reuse pooling indices but instead transfers the entire feature map (at the cost of more memory) to the corresponding decoders and concatenates them to upsampled (via deconvolution) decoder feature maps.
- There is no conv5 and max-pool 5 block in U-Net as in the VGGnet architecture.
- SegNet, on the other hand, uses all of the pre-trained convolutional layer weights from VGG net as pre-trained weights.

Training

- The encoder and decoder weights were all initialized using the technique described in He et al.
- To train all the variants we use stochastic gradient descent (SGD) with a fixed learning rate of 0.1 and momentum of 0.9 using our Caffe implementation of SegNet-Basic.
- We train the variants until the training loss converges.
- Before each epoch, the training set is shuffled and each mini-batch (12 images) is then picked in order thus ensuring that each image is used only once in an epoch. We select the model which performs highest on a validation dataset.
- We use the cross-entropy loss as the objective function for training the network. The loss is summed up over all the pixels in a mini-batch.
- When there is large variation in the number of pixels in each class in the training set then there is a need to weight the loss differently based on the true class. This is termed class balancing.
- We use median frequency balancing where the weight assigned to a class in the loss function is the ratio of the median of class frequencies computed on the entire training set divided by the class frequency.
- This implies that larger classes in the training set have a weight smaller than 1 and the weights of the smallest classes are the highest. We also experimented with training the different variants without class balancing or equivalently using natural frequency balancing.

				Median frequency balancing							Natural frequency balancing						
				Test				Train			Test				Train		
Variant	Params (M)	Storage multiplier	Infer time (ms)	G	C	mIoU	BF	G	C	mIoU	G	C	mIoU	BF	G	C	mIoU
Fixed upsampling																	
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	20.83	89.1	90.2	82.7	82.7	52.5	43.8	23.08	93.5	74.1	59.9
Upsampling using max-pooling indices																	
SegNet-Basic	1.425	1	52.6	82.7	62.0	47.7	35.78	94.7	96. 2	92.7	84.0	54.6	46.3	36.67	96.1	83.9	73.3
SegNet-Basic-EncoderAddition	1.425	64	53.0	83.4	**63.6**	48.5	35.92	94.3	95.8	92.0	**84.2**	56.5	**47.7**	36.27	95.3	80.9	68.9
SegNet-Basic-SingleChannelDecoder	0.625	1	33.1	81.2	60.7	46.1	31.62	93.2	94.8	90.3	83.5	53.9	45.2	32.45	92.6	68.4	52.8
Learning to upsample (bilinear initialisation)																	
FCN-Basic	0.65	11	24.2	81.7	62.4	47.3	**38.11**	92.8	93.6	88.1	83.9	55.6	45.0	**37.33**	92.0	66.8	50.7
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	31.96	92.5	93.0	87.2	82.3	53.9	44.2	29.43	93.1	72.8	57.6
FCN-Basic-NoDimReduction	1.625	64	44.8	**84.1**	63.4	**50.1**	37.37	**95.1**	**96.5**	**93.2**	83.5	**57.3**	47.0	37.13	**97.2**	**91.7**	**84.8**
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	30.47	92.5	94.6	89.9	83.7	54.8	45.5	33.17	95.0	80.2	67.8

TABLE 1

Comparison of decoder variants. We quantify the performance using global (G), class average (C), mean of intersection over union (mIoU) and a semantic contour measure (BF). The testing and training accuracies are shown as percentages for both natural frequency and median frequency balanced training loss function. SegNet-Basic performs at the same level as FCN-Basic but requires only storing max-pooling indices and is therefore more memory efficient during inference. Note that the theoretical memory requirement reported is based only on the size of the first layer encoder feature map. FCN-Basic, SegNet-Basic, SegNet-Basic-EncoderAddition all have high BF scores indicating the need to use information in encoder feature maps for better class contour delineation. Networks with larger decoders and those using the encoder feature maps in full perform best, although they are least efficient in terms of inference time and memory.

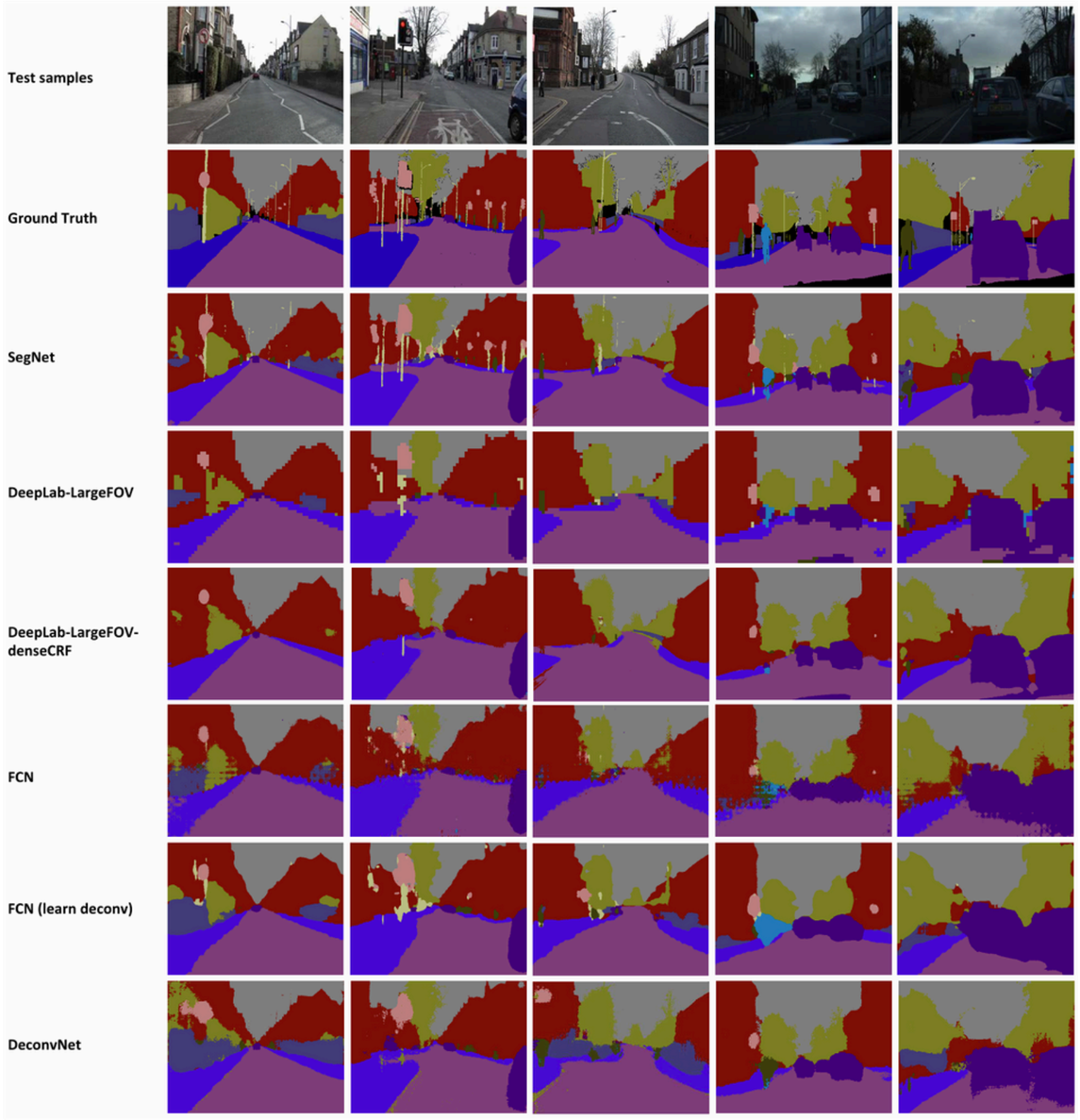


Fig. 4. Results on CamVid day and dusk test samples. SegNet shows superior performance, particularly with its ability to delineate boundaries, as compared to some of the larger models when all are trained in a controlled setting. DeepLab-LargeFOV is the most efficient model and with CRF post-processing can produce competitive results although smaller classes are lost. FCN with learnt deconvolution is clearly better. DeconvNet is the largest model with the longest training time, but its predictions loose small classes. Note that these results correspond to the model corresponding to the highest mIoU accuracy in Table 3.

Network/Iterations	40K				80K				>80K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	88.81	59.93	50.02	35.78	89.68	69.82	57.18	42.08	90.40	71.20	60.10	46.84	140K
DeepLab-LargeFOV [3]	85.95	60.41	50.18	26.25	87.76	62.57	53.34	32.04	88.20	62.53	53.88	32.77	140K
DeepLab-LargeFOV-denseCRF [3]	not computed								89.71	60.67	54.74	40.79	140K
FCN	81.97	54.38	46.59	22.86	82.71	56.22	47.95	24.76	83.27	59.56	49.83	27.99	200K
FCN (learnt deconv) [2]	83.21	56.05	48.68	27.40	83.71	59.64	50.80	31.01	83.14	64.21	51.96	33.18	160K
DeconvNet [4]	85.26	46.40	39.69	27.36	85.19	54.08	43.74	29.33	89.58	70.24	59.77	52.23	260K

TABLE 3

Quantitative comparison of deep networks for semantic segmentation on the CamVid test set when trained on a corpus of 3433 road scenes *without class balancing*. When end-to-end training is performed with the same and fixed learning rate, smaller networks like SegNet learn to perform better in a shorter time. The BF score which measures the accuracy of inter-class boundary delineation is significantly higher for SegNet, DeconvNet as compared to other competing models. DeconvNet matches the metrics for SegNet but at a much larger computational cost. Also see Table 2 for individual class accuracies for SegNet.

U-SegNet

- This paper proposes a Fully Convolutional Neural Network (FCN) tool, which is a hybrid of two widely used deep learning segmentation architectures, SegNet and U-Net, for improved brain tissue segmentation.
- The authors propose a skip connection inspired from U-Net, in the SegNet architetcure, to incorporate fine multiscale information for better tissue boundary identification.
- Since the primary aim of developing this architecture is the task of automating brain tissue segmentation. Let’s look at the challenges associated with it:-
  a. Firstly, there are large variations in brain’s anatomical structures by phenotype such as age, gender, race, and disease. This leads to difficulty in generalizing one specific segmentation method for all phenotypic categories.
  b. Secondly, challenges are associated with the cytoarchitec tural variations such as gyral folds, sulci depths, thin tissue structures, and smooth boundaries between different tissues. This leads to confusion in categorical labeling into distinct tissue classes and is challenging even for a human expert.
  c. Lastly, the imaging technology has its own limitations with reference to bias effects of scanner and, signal-to-noise ratio and motion artifacts in the captured MRI images.

Motivation behind U-SegNet

- The state-of-the-art brain segmentation DL architectures employ 3D models that are computationally heavy and require learning a large number of parameters. This requires a large number of annotated training sample images, while medical imaging data, in general, is limited.
- This work is motivated towards developing a computationally efficient DL technique that works on limited training data and performs brain image segmentation tasks with reasonably good performance.
- The authors explore SegNet and U-Net architectures, which are stated to require much less training data.

U-SegNet: Fully Convolutional Neural Network based Automated Brain tissue segmentation Tool
Pulkit Kumar, Pravin Nagar, Chetan Arora, Anubha Gupta

U-Net and SegNet

- SegNet is a well known architecture in computer vision for semantic segmentation, but has not been used much so far for the brain MRI segmentation task.
- It passes pooling indices to the upsampling layers and hence, requires much fewer parameters and is faster to train.
- U-Net uses multiscale information via skip connections and captures both coarse level and fine level information at the deconvolutional layers.
- However, because of learnable upsampling, U-Net has much larger parameters to learn and is comparatively slower to train than SegNet.
- On the other hand, SegNet does not capture multiscale information as effectively as the U-Net.
- We see the complimentary strengths in the two models and explore a combination of the two in this paper.

Proposed U-SegNet Architecture

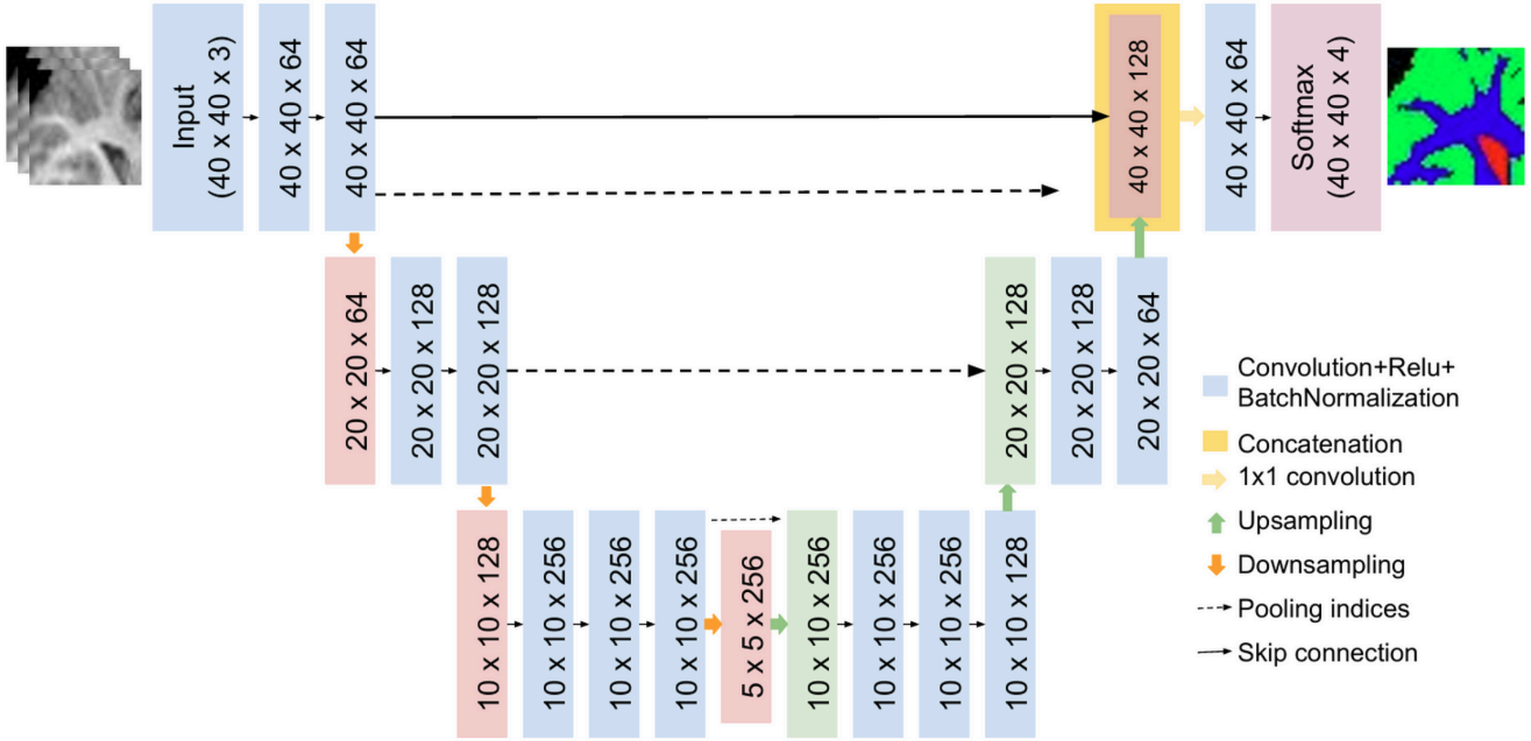


Fig. 1. We propose a Fully Convolutional Neural Network (FCN), that is a hybrid of two widely used deep learning segmentation architectures SegNet and U-Net, for improved brain tissue segmentation. While the base architecture resembles SegNet, we propose a skip connection inspired from U-Net. These skip connection help the proposed network in capturing fine-grained multiscale information for better tissue boundary identification.

Proposed U-SegNet Architecture

- The proposed U-SegNet architecture is a hybrid of both SegNet and U-Net.
- The authors posit that the local information is more important than the global information for identifying WM, GM, and CSF. Hence, a patch-based training is adopted on axial slices of size 256 x 128.
- The authors observed GM structures carefully and noted that for the given resolution IBSR dataset, a patch size of 40 is appropriate to capture sufficient-sized local structures helpful in segmentation.
- With each patch, equal sized patches from the slice above it and below it are concatenated to add 3D volumetric structural context to the segmentation task. Overlapping patches shifted by 10 voxels in both directions on the axial slices are captured for training and testing.

- They have reduced the depth of the SegNet architecture to handle 40 x 40 x 3 sized input patches. Each convolutional layer uses a 3x3 kernel. Max-pool layers of size 2 x 2 and RELU activation functions are used.
- A U-Net type skip connection is introduced only at the uppermost layer to incorporate feature maps with fine details. At this layer, a 1x1 conv. layer is used to consolidate coarser and finer information for the segmentation task and reduce the number of parameters for the final convolutional layer.
- The skip connection helps incorporate fine information without increasing the parameters (as in U-Net).
- In the end, a softmax layer with 4 outputs is used to implement 4-label classification as
    - background (0)
    - GM (1)
    - WM (2)
    - CSF (3).

Dataset Description

- We have used IBSR-18 dataset comprising of 18 T1-weighted MRI volumes of size 256 x 128 x 256 of 4 healthy females and 14 healthy males with age between 7-71 years.
- These volumes are provided after skull-stripping, normalization, and bias field correction.
- The ground truth is provided with man ual segmentation by experts with tissue labels as 0,1,2,3 for background, CSF, GM, and WM, respectively.
- Each MRI volume is read via 256 axial brain slices of size 256 x 128 each in the proposed model.

Implementation Details

- Vanilla SegNet was used with weights initialized from a network trained on the CamVid dataset.
- Training was done sequentially by fine tuning one layer at a time starting from the last layer with low learning rate of 10^(-6).
- Thus, this new architecture is fine-tuned on the SegNet model for the front-end SegNet layers.
- The new convolutional layer in the end receiving information via skip connection and layers afterwards are trained from scratch. Stochastic gradient descent (SGD) optimization, batch size of 64, momentum of 0.9, l2 regularisation with parameter 10^(-4) were used for a maximum epoch of 700 during the training. Theano with Lasagne was used to train all the models.
- While the SegNet architecture has 3475396 learnable parameters, U-Net has 3900996, and the proposed U-SegNet has 3483652 parameters. Thus, there is not a substantial increase in the number of parameters compared to the SegNet architecture.

Training & Test Data

- The authors selected 9 volumes for training and 9 for testing.
- The train-test split comprises all the variation across age and gender.
- At training time, they selected 6 volumes for training and 3 for validation and reported the dice ratio on the test data.
- For testing, the class of a pixel was decided through majority voting of class labels obtained on overlapping patches.
- Dice score (DC) was used as an evaluation metric for all the three tissue classes.

$$DC = \frac{2 * TP}{2 * TP + FP + FN},$$

where TP, FP, and FN represent the true positives, false positives, and false negatives of the class for which the score is calculated.

Results

Models	GM	WM	CSF	Wt. DC
Fuzzy c-means [6]	83.11	**91.83**	21.7	85.13
SegNet [14]	87.36	84.15	59.04	85.92
U-Net [15]	86.87	83.58	58.36	85.40
Proposed U-SegNet	**90.33**	89.23	**66.58**	**89.64**
Proposed U-SegNet-2 with two skip connections	88.17	85.95	57.81	87.03

Table 1. Dice ratio comparison of our method with state-of-the-art approaches.

- SegNet tends to miss out the finer details especially at the boundary between white matter and gray matter. U-Net, on the other hand, because of the skip connections from the lower levels, is able to capture the fine details, say, at the boundaries more accurately than SegNet.
- However, U-Net gives errors at places where one class is present in abundance. Also, we observe random noise in the U-Net-based segmentation, which we speculate to be because of the confusion created by the deconvolutional layers and skip connections at the lower levels.
- U-SegNet incorporates the good features of both U-Net as well as SegNet.

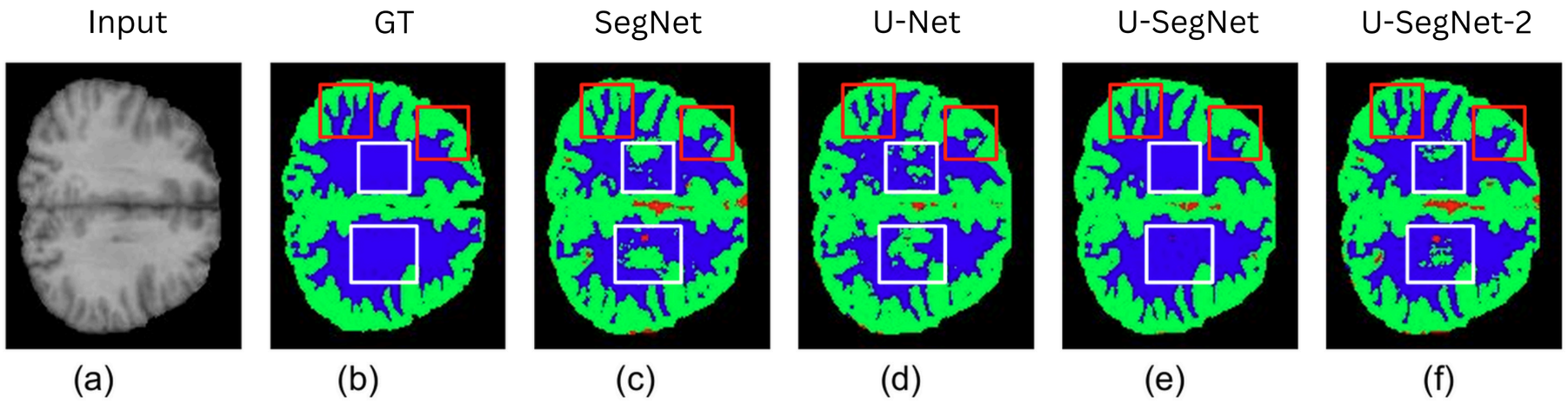


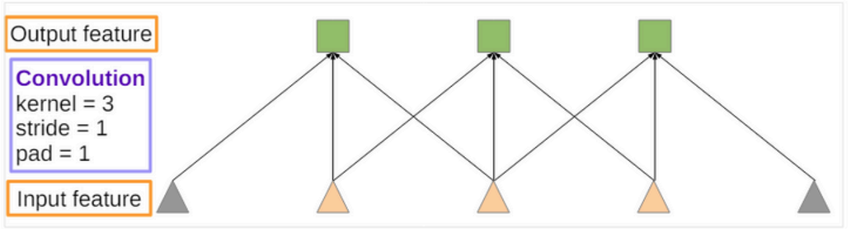
Fig. 2. Visualization of (a) Input, (b) Ground Truth labeled image; Black: background, Green: GM, Blue: WM, Red: CSF (c) SegNet, (d) U-Net, (e) Proposed U-SegNet architecture with one skip connection, and (f) Proposed U-SegNet architecture with two skip connections (U-SegNet-2). It is observed that SegNet and U-Net show random patches (white rectangles) and compromise fine details (red rectangles) around the folds generated by gyri and sulci. The proposed architecture captures fine details and solve the random noise problem in the continuous WM seen in U-Net and SegNet. We further observe that adding one more skip connection at the second level in (f) leads to loss of fine information and also fails to handle random noise present in the continuous WM. SegNet, U-Net, and the proposed U-SegNet architecture with two skip connections show that adding skip connections at lower layers leads to performance drop.

DeepLab

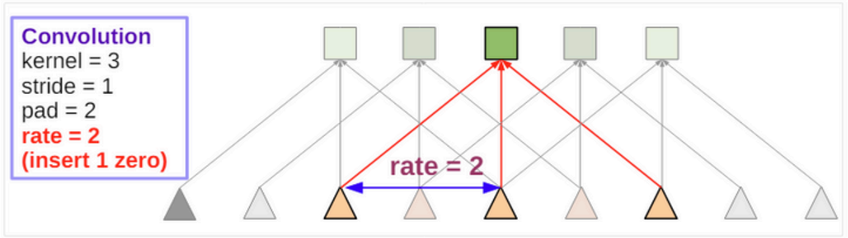
- Deep Convolutional Neural Networks (DCNNs) have pushed the performance of computer vision systems to great heights on a broad array of high-level problems, including image classification and object detection.
- The built-in invariance of DCNNs to local image transformations, which allows them to learn increasingly abstract data representations is essential to their success.
- This invariance is desirable for classification tasks, but can hamper dense prediction tasks such as semantic segmentation, where abstraction of spatial information is undesired.
- The authors consider three challenges in the application of DCNNs to semantic image segmentation:
  - reduced feature resolution
  - existence of objects at multiple scales
  - reduced localization accuracy due to DCNN invariance

DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs
Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille

- The first challenge is caused by the repeated combination of max-pooling and downsampling (‘striding’) performed at consecutive layers of DCNNs originally designed for image classification.
- This results in feature maps with significantly reduced spatial resolution when the DCNN is employed in a fully convolutional fashion.
- To overcome this hurdle and efficiently produce denser feature maps, we remove the downsampling operator from the last few max pooling layers of DCNNs. Instead, we upsample the filters in subsequent convolutional layers, resulting in feature maps computed at a higher sampling rate.
- Filter upsampling amounts to inserting holes (‘trous’ in French) between nonzero filter taps. We use the term atrous convolution as a shorthand for convolution with upsampled filters.
- In practice, we recover full-resolution feature maps by a combination of atrous convolution, which computes feature maps more densely, followed by simple bilinear interpolation of the feature responses to the original image size.
- This scheme offers a simple yet powerful alternative to using deconvolutional layers in dense prediction tasks.
- Compared to regular convolution with larger filters, atrous convolution allows us to effectively enlarge the field of view of filters without increasing the number of parameters or the amount of computation.



(a) Sparse feature extraction



(b) Dense feature extraction

Fig. 2: Illustration of atrous convolution in 1-D. (a) Sparse feature extraction with standard convolution on a low resolution input feature map. (b) Dense feature extraction with atrous convolution with rate $r = 2$, applied on a high resolution input feature map.

- The second challenge is caused by the existence of objects at multiple scales.
- A standard way to deal with this is to present to the DCNN rescaled versions of the same image and then aggregate the feature or score maps.
- We show that this approach indeed increases the performance of our system but comes at the cost of computing feature responses at all DCNN layers for multiple scaled versions of the input image.
- Instead, motivated by spatial pyramid pooling we propose a computationally efficient scheme of resampling a given feature layer at multiple rates prior to convolution.
- This amounts to probing the original image with multiple filters that have complementary effective fields of view, thus capturing objects as well as useful image context at multiple scales.
- Rather than actually resampling features, we efficiently implement this mapping using multiple parallel atrous convolutional layers with different sampling rates; we call the proposed technique “atrous spatial pyramid pooling” (ASPP).

- The third challenge relates to the fact that an object centric classifier requires invariance to spatial transformations, inherently limiting the spatial accuracy of a DCNN.
- One way to mitigate this problem is to use skip-layers to extract “hyper-column” features from multiple network layers when computing the final segmentation result.
- Our work explores an alternative approach which is highly effective. In particular, we boost our model’s ability to capture fine details by employing a fully-connected Conditional Random Field (CRF).
- CRFs have been broadly used in semantic segmentation to combine class scores computed by multi-way classifiers with the low-level information captured by the local interactions of pixels and edges or superpixels.
- We use the fully connected pairwise CRF proposed by for its efficient computation, and ability to capture fine edge details while also catering for long range dependencies.
- We demonstrate that it leads to state-of-the-art results when coupled with a DCNN-based pixel-level classifier.

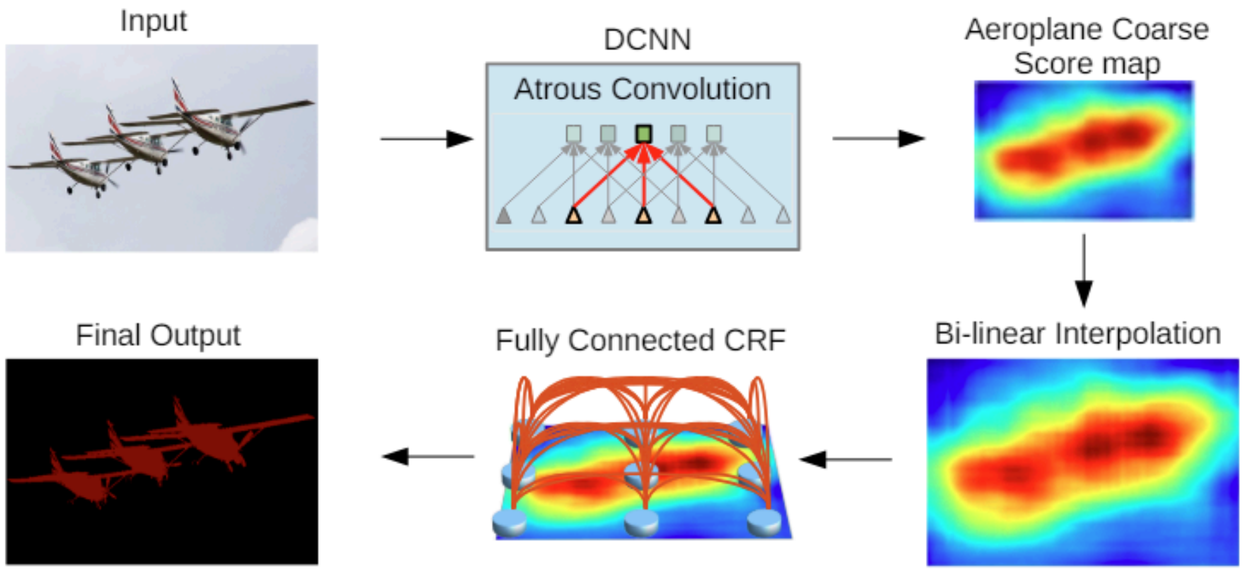


Fig. 1: Model Illustration. A Deep Convolutional Neural Network such as VGG-16 or ResNet-101 is employed in a fully convolutional fashion, using atrous convolution to reduce the degree of signal downsampling (from 32x down 8x). A bilinear interpolation stage enlarges the feature maps to the original image resolution. A fully connected CRF is then applied to refine the segmentation result and better capture the object boundaries.

Atrous Convolutions

- Considering one-dimensional signals first, the output y[i] of atrous convolution of a 1-D input signal x[i] with a filter w[k] of length K is defined as:

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k].$$

- The rate parameter r corresponds to the stride with which we sample the input signal. Standard convolution is a special case for rate r = 1.

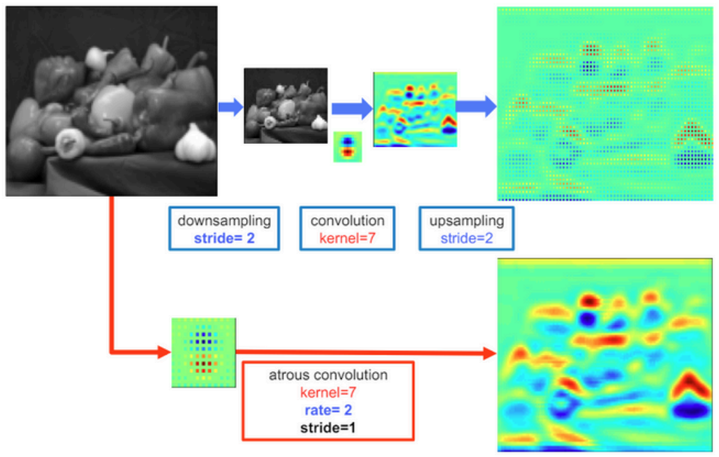
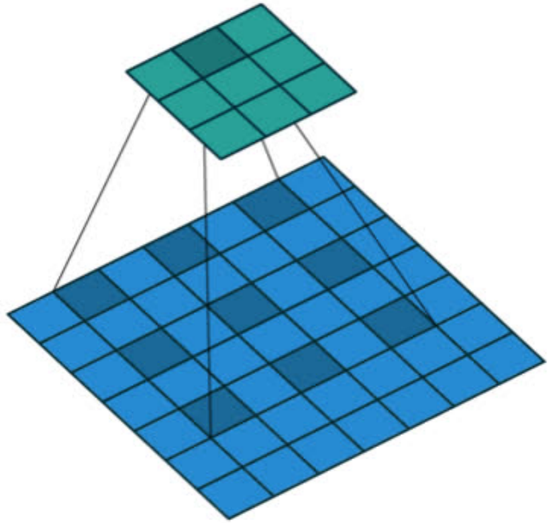


Fig. 3: Illustration of atrous convolution in 2-D. Top row: sparse feature extraction with standard convolution on a low resolution input feature map. Bottom row: Dense feature extraction with atrous convolution with rate $r = 2$, applied on a high resolution input feature map.

- Given an image, we assume that we first have a downsampling operation that reduces the resolution by a factor of 2, and then perform a convolution with a kernel - here, the vertical Gaussian derivative.
- If one implants the resulting feature map in the original image coordinates, we realize that we have obtained responses at only 1/4 of the image positions.
- Instead, we can compute responses at all image positions if we convolve the full resolution image with a filter ‘with holes’, in which we upsample the original filter by a factor of 2, and introduce zeros in between filter values.
- Although the effective filter size increases, we only need to take into account the non-zero filter values, hence both the number of filter parameters and the number of operations per position stay constant.
- The resulting scheme allows us to easily and explicitly control the spatial resolution of neural network feature responses.



GIF source

Atrous Spatial Pyramid Pooling (ASPP)

- The first approach to handling scale variablity in semantic segmentation amounts to standard multiscale processing.
- We extract DCNN score maps from multiple (three in the paper’s experiments) rescaled versions of the original image using parallel DCNN branches that share the same parameters.
- To produce the final result, we bilinearly interpolate the feature maps from the parallel DCNN branches to the original image resolution and fuse them, by taking at each position the maximum response across the different scales.
- We do this both during training and testing. Multiscale processing significantly improves performance but at the cost of computing feature responses at all DCNN layers for multiple scales of input.

- The second approach is inspired by the success of the R-CNN spatial pyramid pooling method of He et al., which showed that regions of an arbitrary scale can be accurately and efficiently classified by resampling convolutional features extracted at a single scale.
- The authors have implemented a variant of their scheme that uses multiple parallel atrous convolutional layers with different sampling rates.
- The features extracted for each sampling rate are further processed in separate branches and fused to generate the final result.
- The proposed “atrous spatial pyramid pooling” (DeepLab-ASPP) approach generalizes the DeepLab-LargeFOV variant.

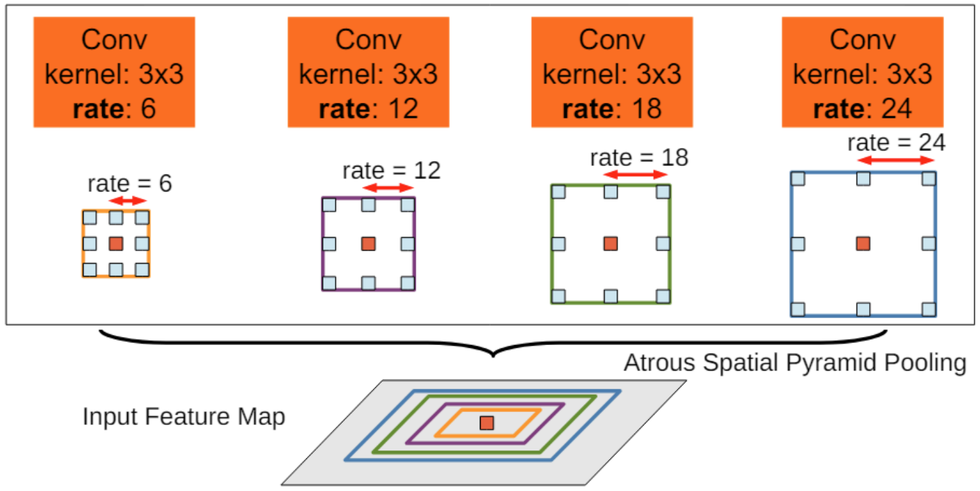


Fig. 4: Atrous Spatial Pyramid Pooling (ASPP). To classify the center pixel (orange), ASPP exploits multi-scale features by employing multiple parallel filters with different rates. The effective Field-Of-Views are shown in different colors.

Structured Prediction with Fully-Connected Conditional Random Fields for Accurate Boundary Recovery

- Previous work has pursued two directions to address this localization challenge.
  - The first approach is to harness information from multiple layers in the convolutional network in order to better estimate the object boundaries.
  - The second is to employ a super-pixel representation, essentially delegating the localization task to a low-level segmentation method.
- The authors pursue an alternative direction based on coupling the recognition capacity of DCNNs and the fine-grained localization accuracy of fully connected CRFs.
- Traditionally, conditional random fields (CRFs) have been employed to smooth noisy segmentation maps.
- Typically these models couple neighboring nodes, favoring same-label assignments to spatially proximal pixels.
- Qualitatively, the primary function of these short-range CRFs is to clean up the spurious predictions of weak classifiers built on top of local hand-engineered features.

- Compared to these weaker classifiers, modern DCNN architectures such as the one used in this work produce score maps and semantic label predictions which are qualitatively different.
- The score maps are typically quite smooth and produce homogeneous classification results.
- In this regime, using short-range CRFs can be detrimental, as our goal should be to recover detailed local structure rather than further smooth it.
- To overcome these limitations of short-range CRFs, we integrate into our system the fully connected CRF model of P. Krahenbuhl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials”.
- The model employs the energy function:-

$$E(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)$$

where $\boldsymbol{x}$ is the label assignment for pixels. We use as unary potential $\theta_i(x_i) = -\log P(x_i)$, where $P(x_i)$ is the label assignment probability at pixel i as computed by a DCNN.

- The pairwise potential has a form that allows for efficient inference while using a fully-connected graph, i.e. when connecting all pairs of image pixels, i, j. The following expression is used:-

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) + w_2 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\gamma^2} \right) \right] \quad (3)$$

- Here, μ(xi,xj) = 1 if xi not equal to xj , and zero otherwise, which, as in the Potts model, means that only nodes with distinct labels are penalized.
- The remaining expression uses two Gaussian kernels in different feature spaces;
  - The first, ‘bilateral’ kernel depends on pixel positions (p) and RGB color (I). It forces pixels with similar color and position to have similar labels.
  - The second kernel only depends on pixel positions. It only considers spatial proximity when enforcing smoothness.
- The hyper parameters σα, σβ and σγ control the scale of Gaussian kernels.

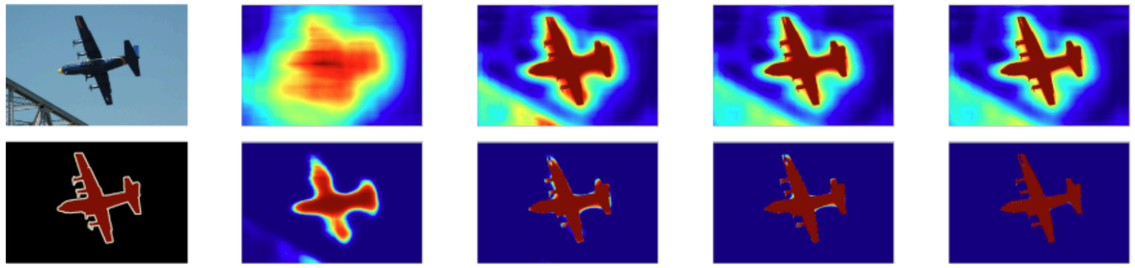


Fig. 5: Score map (input before softmax function) and belief map (output of softmax function) for Aeroplane. We show the score (1st row) and belief (2nd row) maps after each mean field iteration. The output of last DCNN layer is used as input to the mean field inference.

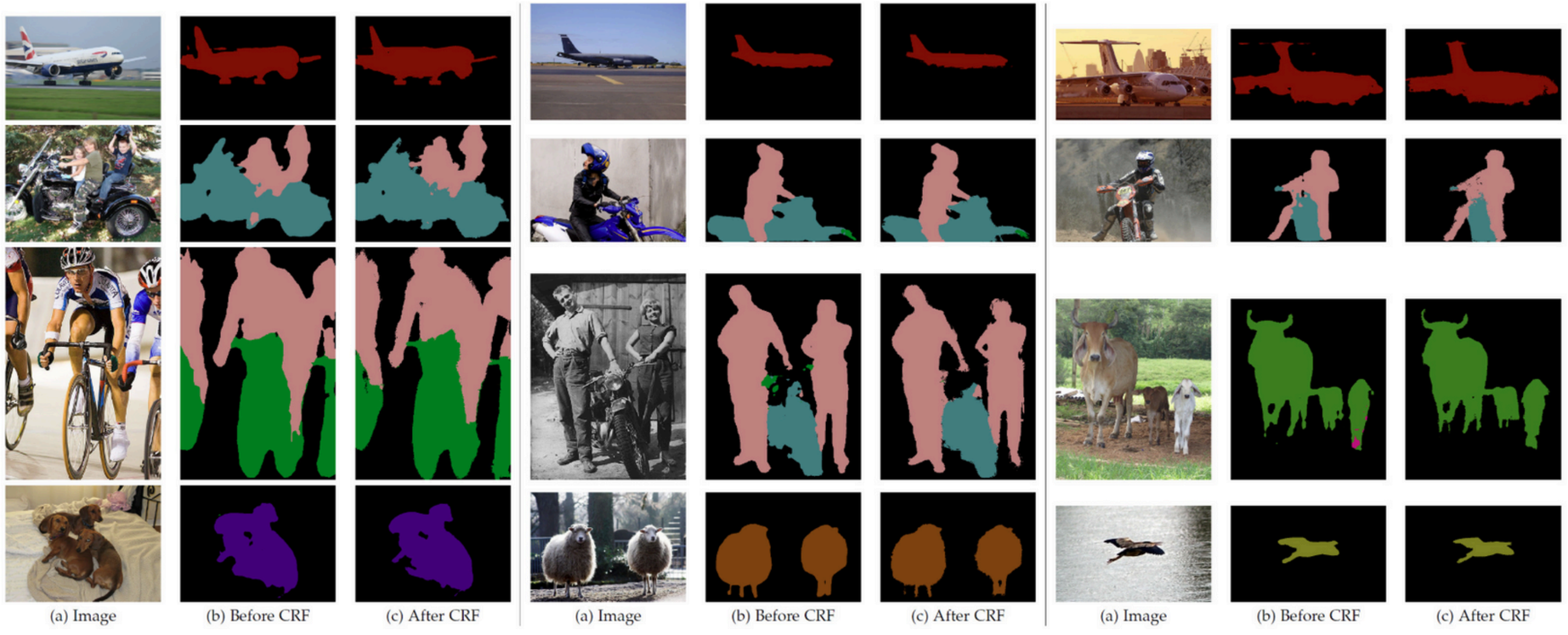
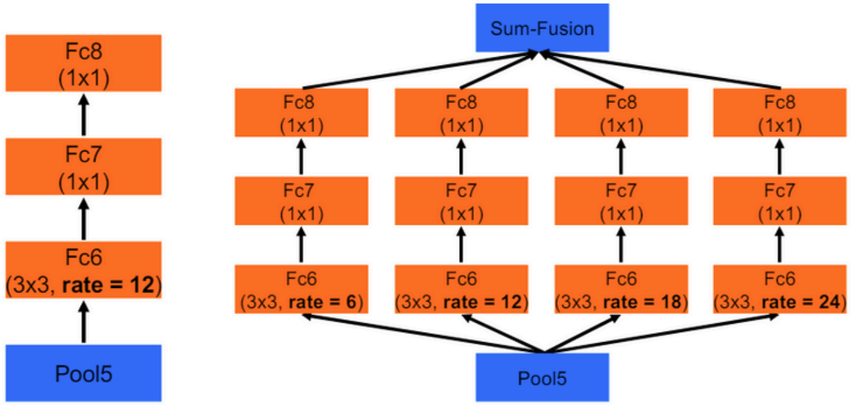
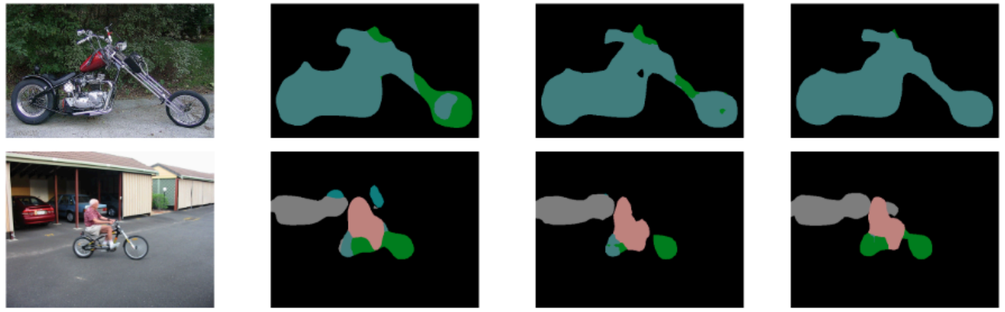


Fig. 6: PASCAL VOC 2012 *val* results. Input image and our DeepLab results before/after CRF.



(a) DeepLab-LargeFOV (b) DeepLab-ASPP

Fig. 7: DeepLab-ASPP employs multiple filters with different rates to capture objects and context at multiple scales.



(a) Image (b) LargeFOV (c) ASPP-S (d) ASPP-L

Fig. 8: Qualitative segmentation results with ASPP compared to the baseline LargeFOV model. The **ASPP-L** model, employing multiple *large* FOVs can successfully capture objects as well as image context at multiple scales.

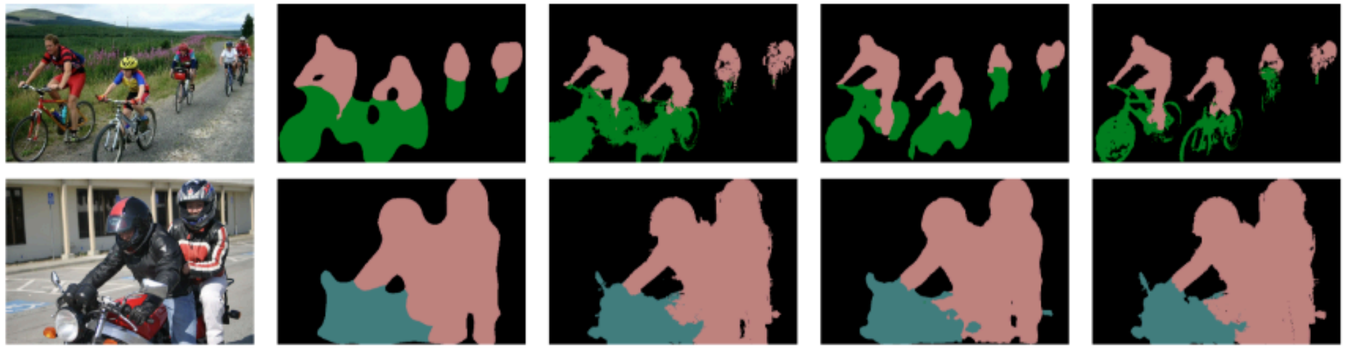


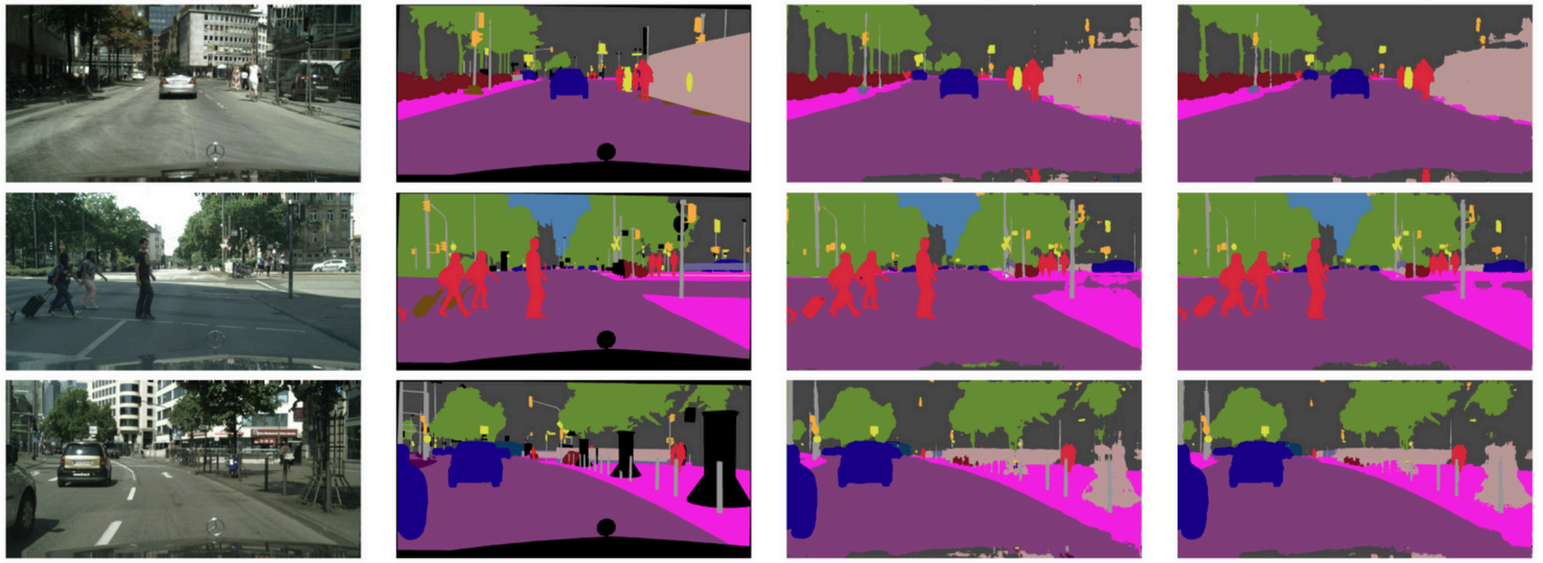
Fig. 9: DeepLab results based on VGG-16 net or ResNet-101 before and after CRF. The CRF is critical for accurate prediction along object boundaries with VGG-16, whereas ResNet-101 has acceptable performance even before CRF.

Method	mIOU
DeepLab-CRF-LargeFOV-COCO [58]	72.7
MERL_DEEP_GCRF [88]	73.2
CRF-RNN [59]	74.7
POSTECH_DeconvNet_CRF_VOC [61]	74.8
BoxSup [60]	75.2
Context + CRF-RNN [76]	75.3
QO_4^{mres} [66]	75.5
DeepLab-CRF-Attention [17]	75.7
CentraleSuperBoundaries++ [18]	76.0
DeepLab-CRF-Attention-DT [63]	76.3
H-ReNet + DenseCRF [89]	76.8
LRR_4x_COCO [90]	76.8
DPN [62]	77.5
Adelaide_Context [40]	77.8
Oxford_TVG_HO_CRF [91]	77.9
Context CRF + Guidance CRF [92]	78.1
Adelaide_VeryDeep_FCN_VOC [93]	79.1
DeepLab-CRF (ResNet-101)	79.7

TABLE 5: Performance on PASCAL VOC 2012 *test* set. We have added some results from recent arXiv papers on top of the official leadearboard results.

Method	MSC	COCO	Aug	LargeFOV	ASPP	CRF	mIOU
VGG-16							
DeepLab [38]				✓			37.6
DeepLab [38]				✓		✓	39.6
ResNet-101							
DeepLab							39.6
DeepLab	✓		✓				41.4
DeepLab	✓	✓	✓				42.9
DeepLab	✓	✓	✓	✓			43.5
DeepLab	✓	✓	✓		✓		44.7
DeepLab	✓	✓	✓		✓	✓	45.7
O_2P [45]							18.1
CFM [51]							34.4
FCN-8s [14]							37.8
CRF-RNN [59]							39.3
ParseNet [86]							40.4
BoxSup [60]							40.5
HO_CRF [91]							41.3
Context [40]							43.3
VeryDeep [93]							44.5

TABLE 6: Comparison with other state-of-art methods on PASCAL-Context dataset.

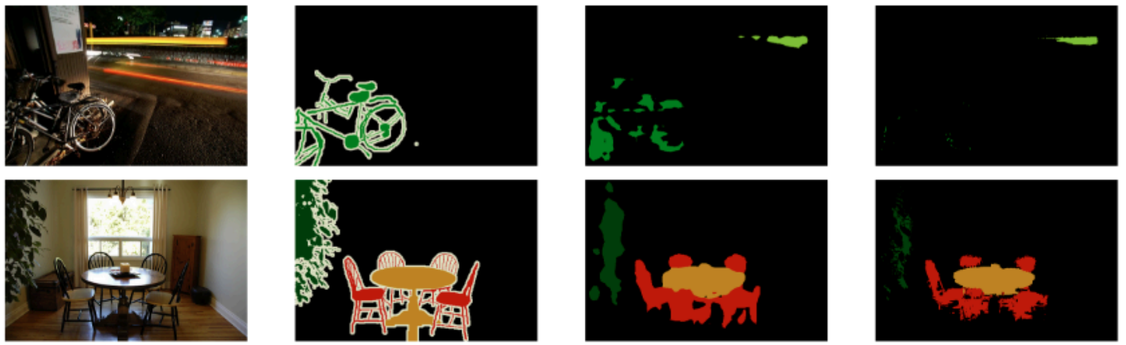


(a) Image (b) G.T. (c) Before CRF (d) After CRF

Fig. 13: Cityscapes results. Input image, ground-truth, and our DeepLab results before/after CRF.

Full	Aug	LargeFOV	ASPP	CRF	mIOU
VGG-16					
		✓			62.97
		✓		✓	64.18
✓		✓			64.89
✓		✓		✓	65.94
ResNet-101					
✓					66.6
✓		✓			69.2
✓			✓		70.4
✓	✓		✓		71.0
✓	✓		✓	✓	71.4

TABLE 9: Val set results on Cityscapes dataset. **Full**: model trained with full resolution images.



(a) Image (b) G.T. (c) Before CRF (d) After CRF

Fig. 14: Failure modes. Input image, ground-truth, and our DeepLab results before/after CRF.

Thank You